Amazon Elastic Container Service

开发人员指南

API 版本 2014-11-13

Amazon Elastic Container Service: 开发人员指南

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

什么是 Amazon ECS？ 1
- Amazon ECS 的功能 1
  - 容器和映像 2
  - 任务定义 3
  - 任务和计划 3
  - 集群 4
  - 容器代理 4
- 如何开始使用 Amazon ECS 5
- 相关服务 5
- 访问 Amazon ECS 5

设置 7
- 注册 AWS 7
- 创建 IAM 用户 7
- 创建 IAM 角色 8
- 创建密钥对 9
- 创建 Virtual Private Cloud 11
- 创建安全组 12
- 安装 AWS CLI 13

Amazon ECS 的 Docker 基本知识 14
- 安装 Docker 14
- 创建 Docker 映像 15
- (可选) 将您的映像推送至 Amazon Elastic Container Registry 16
- （可选）清除 17
- 后续步骤 17

开始使用 Amazon ECS 19
- 通过 Fargate 开始使用 Amazon ECS 19
  - 先决条件 19
  - 步骤 1：创建任务定义 20
  - 步骤 2：配置服务 20
  - 步骤 3：配置集群 21
  - 步骤 4: 检查 21
  - 步骤 5：（可选）查看您的服务 21
- 开始使用 Amazon ECS 22
  - 先决条件 22
  - 步骤 1：选择您的配置选项 22
  - 步骤 2：创建任务定义 23
  - 步骤 3：配置服务 24
  - 步骤 4：配置集群 24
  - 步骤 5: 审核 25
  - 步骤 6：（可选）查看您的服务 25
- 清除 26
  - 缩减服务 26
  - 删除服务 26
  - 取消注册容器实例 26
  - 删除集群 26
  - 删除 AWS CloudFormation 堆栈 27

AWS Fargate 28
- 任务定义 28
  - 网络模式 29
  - 任务 CPU 和内存 29
  - 日志记录 30
  - Amazon ECS 任务执行 IAM 角色 30
  - 示例任务定义 30
  - 任务存储 31

任务和服务 32
任务联网 32
私有注册表身份验证 32
集群 33
Fargate 任务停用 33
平台版本 33
平台版本注意事项 33
可用的平台版本 33
在 Amazon ECS 上使用 GPU 35
有关使用 GPU 的注意事项 35
在任务定义中指定 GPU 36
集群 38
创建集群 38
扩展集群 40
删除集群 41
任务定义 43
应用程序架构 43
使用 Fargate 启动类型 43
使用 EC2 启动类型 44
创建任务定义 45
任务定义模板 48
任务定义参数 52
系列 52
任务角色 52
任务执行角色 52
网络模式 52
容器定义 53
卷 74
任务放置约束 76
启动类型 77
任务大小 77
其他任务定义参数 78
启动类型 81
Fargate 启动类型 81
EC2 启动类型 82
在任务中使用数据卷 83
Fargate 任务存储 84
Docker 卷 85
绑定挂载 89
任务联网 95
任务联网方面的考量 96
启用任务联网 97
使用 awslogs 日志驱动程序 97
在容器上启用 awslogs 日志驱动程序 98
创建您的日志组 98
可用的 awslogs 日志驱动程序选项 99
在任务定义中指定日志配置 101
在 CloudWatch Logs 中查看 awslogs 容器日志 102
任务的私有注册表身份验证 104
私有注册表身份验证所需的 IAM 权限 105
启用私有注册表身份验证 105
指定敏感数据 107
有关指定敏感数据的注意事项 107
作为环境变量注入敏感数据 107
注入日志配置中的敏感数据 108
Amazon ECS 密钥所需的 IAM 权限 109
创建 AWS Secrets Manager 密钥 109

创建 AWS Systems Manager Parameter Store 参数 110
创建引用密钥的任务定义 110
示例任务定义 112
示例：Web 服务器 112
示例：WordPress 和 MySQL 113
示例：`awslogs` 日志驱动程序 114
示例：`splunk` 日志驱动程序 114
示例：`fluentd` 日志驱动程序 115
示例：`gelf` 日志驱动程序 115
示例：Amazon ECR 映像和任务定义 IAM 角色 116
示例：带命令的入口点 116
示例：容器依赖项 116
更新任务定义 118
取消注册任务定义 118
账户设置 120
Amazon 资源名称 (ARN) 和 ID 121
查看账户设置 122
修改账户设置 123
容器实例 125
容器实例概念 125
容器实例生命周期 126
检查您的账户的实例角色 126
Amazon ECS-optimized AMI 127
如何启动最新的 Amazon ECS-Optimized Amazon Linux AMI 137
AMI 版本 146
AMI 存储配置 151
检索 Amazon ECS-Optimized AMI 元数据 155
订阅Amazon ECS-Optimized Amazon Linux AMI 更新通知 161
Amazon SNS 消息格式 163
启动容器实例 165
引导启动容器实例 168
Amazon ECS 容器代理 168
Docker 守护程序 169
cloud-init-per 实用工具 169
使用 MIME 分段存档指定多个用户数据块 170
示例用户数据脚本 171
弹性网络接口中继 175
ENI 中继注意事项 175
使用具有增加的 ENI 限制的容器实例 176
支持的 Amazon EC2 实例类型 178
连接到您的容器实例 180
CloudWatch Logs 181
CloudWatch Logs IAM 策略 182
安装 CloudWatch Logs 代理 182
配置并启动 CloudWatch Logs 代理 183
查看 CloudWatch Logs 185
在启动时使用用户数据配置 CloudWatch Logs 186
容器实例耗尽 190
耗尽实例 190
内存管理 190
预留系统内存 191
查看容器实例内存 191
远程管理容器实例 192
Run Command IAM 策略 193
在Amazon ECS-Optimized AMI 上安装 SSM 代理 193
使用 Run Command 194
在容器实例启动时启动任务 194

取消注册容器实例 196
容器代理 198
安装 Amazon ECS 容器代理 198
在 Amazon Linux 2 EC2 实例上安装 Amazon ECS 容器代理 198
在 Amazon Linux AMI EC2 实例上安装 Amazon ECS 容器代理 199
在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理 200
使用主机网络模式运行 Amazon ECS 容器代理 207
容器代理版本 207
Amazon ECS-Optimized Amazon Linux 2 AMI 容器代理版本 207
Amazon ECS-Optimized Amazon Linux AMI 容器代理版本 208
更新 Amazon ECS 容器代理 211
检查 Amazon ECS 容器代理版本 211
更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理 213
手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI） 215
Amazon ECS 容器代理配置 217
可用参数 217
将容器实例配置存储在 Amazon S3 中 227
容器实例的私有注册表身份验证 228
身份验证格式 228
启用私有注册表 230
自动化任务和映像清除 231
可调节参数 231
清除工作流程 232
容器元数据文件 232
启用容器元数据 233
容器元数据文件位置 233
容器元数据文件格式 234
任务元数据终端节点 236
任务元数据终端节点版本 3 236
任务元数据终端节点版本 2 240
Amazon ECS 容器代理自检 244
HTTP 代理配置 245
Amazon Linux 容器实例配置 245
Windows 容器实例配置 248
计划任务 250
正在运行的任务 251
运行使用 Fargate 启动类型的任务 251
运行使用 EC2 启动类型的任务 253
任务放置 255
任务放置策略 255
任务放置约束 257
集群查询语言 260
排定的任务 (cron) 263
任务生命周期 265
生命周期状态 266
任务停用 267
使用计划停用的任务 267
Fargate 任务回收 268
使用 AWS CLI 创建计划任务 268
服务 270
服务计划程序概念 270
守护程序 270
副本 271
附加服务概念 271
服务定义参数 272
部署类型 277
滚动更新 278

使用 CodeDeploy 进行蓝/绿部署 278
外部部署 281
服务负载均衡 284
负载均衡概念 284
负载均衡器类型 285
检查您的账户的服务角色 288
创建负载均衡器 289
向服务注册多个目标组 298
服务 Auto Scaling 299
服务 Auto Scaling 需要 IAM 权限 300
目标跟踪扩展策略 300
步进扩展策略 305
服务发现 306
服务发现 概念 306
服务发现 注意事项 307
Amazon ECS 控制台体验 308
服务发现 定价 308
教程：使用 服务发现 创建服务 308
创建服务 317
步骤 1：配置基本服务参数 317
步骤 2：配置网络 319
步骤 3：将您的服务配置为使用负载均衡器 319
步骤 4：（可选）将服务配置为使用服务发现 323
步骤 5：（可选）将服务配置为使用服务 Auto Scaling 324
步骤 6：查看并创建您的服务 326
更新服务 326
删除服务 328
服务限制逻辑 329
资源和标签 330
标记您的资源 330
有关标签的基本知识 330
标记您的资源 331
标签限制 331
标记资源以便于计费 332
通过控制台使用标签 332
通过 CLI 或 API 使用标签 334
使用率报告 335
接口 VPC 终端节点 (AWS PrivateLink) 337
Amazon ECS VPC 终端节点注意事项 337
为 Amazon ECS 创建 VPC 终端节点 337
存储库 339
在 Amazon ECS 中使用 Amazon ECR 映像 339
监控 340
监控工具 340
自动化工具 340
手动工具 341
CloudWatch 指标 341
启用 CloudWatch 指标 342
可用指标和维度 342
集群预留 345
集群利用率 346
服务利用率 347
服务 RUNNING 任务计数 347
查看 Amazon ECS 指标 348
教程：通过 CloudWatch 警报扩展 349
CloudWatch Events 354
Amazon ECS 事件 354

处理事件 ... 359
教程：侦听 Amazon ECS CloudWatch Events ... 360
教程：发送任务已停止事件的 Amazon Simple Notification Service 警报 ... 362
CloudWatch Container Insights ... 363
使用已启用 Container Insights 的集群 ... 364
使用 Amazon ECS 记录 AWS CloudTrail API 调用 ... 365
CloudTrail 中的 Amazon ECS 信息 ... 365
了解 Amazon ECS 日志文件条目 ... 365
安全性 ... 367
Identity and Access Management ... 367
受众 ... 368
使用身份进行身份验证 ... 368
使用策略管理访问 ... 369
Amazon Elastic Container Service 如何与 IAM 协同工作 ... 370
基于身份的策略示例 ... 374
支持的资源级权限 ... 385
使用服务相关角色 ... 392
托管策略和信任关系 ... 396
Amazon ECS 容器实例 IAM 角色 ... 404
Amazon ECS 任务执行 IAM 角色 ... 406
Amazon ECS 服务计划程序 IAM 角色 ... 410
Amazon ECS CodeDeploy IAM 角色 ... 411
Amazon ECS 服务 Auto Scaling IAM 角色 ... 414
CloudWatch Events IAM 角色 ... 416
Amazon ECS 任务角色 ... 418
任务的 IAM 角色 ... 418
故障排除 ... 422
使用 Amazon ECS CLI ... 425
安装 Amazon ECS CLI ... 425
步骤 1：下载 Amazon ECS CLI ... 425
步骤 2：(可选) 验证 Amazon ECS CLI ... 425
步骤 3：将执行权限应用于二进制文件 ... 430
步骤 4：完成安装 ... 430
配置 Amazon ECS CLI ... 430
配置文件 ... 431
集群配置 ... 431
优先顺序 ... 431
迁移配置文件 ... 432
将较旧的配置文件迁移到 v1.0.0+ 格式 ... 432
教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群 ... 433
先决条件 ... 433
步骤 1：创建任务执行 IAM 角色 ... 433
步骤 2：配置 Amazon ECS CLI ... 434
步骤 3：创建集群和安全组 ... 434
步骤 4：创建复合文件 ... 434
步骤 5：将复合文件部署到集群 ... 435
步骤 6：查看集群上正在运行的容器 ... 435
步骤 7：查看容器日志 ... 436
步骤 8：在集群上扩展任务 ... 436
步骤 9：（可选）查看您的 Web 应用程序 ... 435
步骤 10：清除 ... 437
教程：使用 Amazon ECS CLI 创建包含 EC2 任务的集群 ... 437
先决条件 ... 437
步骤 1：配置 Amazon ECS CLI ... 437
步骤 2：创建您的集群 ... 438
步骤 3：创建复合文件 ... 438
步骤 4：将复合文件部署到集群 ... 441

步骤 5：查看集群上正在运行的容器 441
步骤 6：在集群上扩展任务 441
步骤 7：从复合文件创建 ECS 服务 442
步骤 8：清除 442
教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 442
先决条件 442
配置 Amazon ECS CLI 442
创建一个配置为使用服务发现的 Amazon ECS 服务 443
Amazon ECS 命令行参考 445
ecs-cli 445
ecs-cli configure 447
ecs-cli up 453
ecs-cli down 460
ecs-cli scale 462
ecs-cli ps 463
ecs-cli push 465
ecs-cli pull 467
ecs-cli images 468
ecs-cli license 471
ecs-cli compose 472
ecs-cli compose service 482
ecs-cli logs 503
ecs-cli check-attributes 505
ecs-cli registry-creds 506
ecs-cli local 511
使用 Docker 复合文件语法 518
使用 Amazon ECS 参数 520
使用 AWS CLI 525
教程：使用 AWS CLI 创建包含 Fargate 任务的集群 525
先决条件 525
第 1 步：（可选）创建集群 526
第 2 步：注册任务定义 526
第 3 部：列出任务定义 528
步骤 4：创建服务 528
步骤 5：列出服务 529
步骤 6：描述正在运行的服务 530
教程：使用 AWS CLI 创建包含 EC2 任务的集群 531
先决条件 531
第 1 步：（可选）创建集群 532
第 2 步：使用 Amazon ECS AMI 启动实例 532
第 3 步：列出容器实例 533
第 4 步：描述容器实例 534
第 5 步：注册任务定义 536
第 6 部：列出任务定义 537
第 7 步：运行任务 537
第 8 步：列出任务 538
第 9 步：描述正在运行的任务 538
服务限制 540
常见使用案例 541
微服务 541
Auto Scaling 541
服务发现 541
授权和密钥管理 542
日志记录 542
持续集成和持续部署 542
批处理作业 542
App Mesh 和 Amazon ECS 544

先决条件 544
步骤 1：创建您的服务网格 544
步骤 2：创建您的虚拟节点 544
步骤 3：创建您的虚拟路由器 545
步骤 4：创建您的路由 546
步骤 5：创建您的虚拟服务 546
Update Your Microservice Task Definitions 547
Proxy Configuration 547
Application Container Envoy Dependency 548
Envoy Container Definition 548
Credentials 549
Update an Existing Task Definition 549
Example Task Definitions 550
Amazon ECS 上的 AWS Deep Learning Containers 554
教程 555
Tutorial: Creating a VPC 555
Step 1: Create an Elastic IP Address for Your NAT Gateway 555
Step 2: Run the VPC Wizard 555
Step 3: Create Additional Subnets 556
Next Steps 556
教程：使用 Secrets Manager 密钥指定敏感数据 557
先决条件 557
步骤 1：创建 Secrets Manager 密钥 557
步骤 2：更新任务执行 IAM 角色 558
步骤 3：创建 Amazon ECS 任务定义 558
步骤 4：创建 Amazon ECS 集群 559
步骤 5：运行 Amazon ECS 任务 560
步骤 6：验证 560
步骤 7：清除 561
教程：使用蓝/绿部署创建服务 561
先决条件 562
第 1 步：创建应用程序负载均衡器 562
步骤 2：创建 Amazon ECS 集群 563
第 3 步：注册任务定义 563
步骤 4：创建 Amazon ECS 服务 564
步骤 5：创建 AWS CodeDeploy 资源 564
步骤 5：创建和监控 CodeDeploy 部署 566
步骤 6：清除 568
教程：CodePipeline 持续部署 569
先决条件 569
步骤 1：将构建规范文件添加至您的源存储库 570
步骤 2：创建持续部署管道 571
步骤 3：向 CodeBuild 角色添加 Amazon ECR 权限 572
步骤 4：测试您的管道 572
Tutorial: Using Amazon EFS 573
Step 1: Gather Cluster Information 574
Step 2: Create a Security Group for an Amazon EFS File System 574
Step 3: Create an Amazon EFS File System 574
Step 4: Configure Container Instances 575
Step 5: Create a Task Definition to Use the Amazon EFS File System 577
Step 6: Add Content to the Amazon EFS File System 578
Step 7: Run a Task and View the Results 578
故障排除 579
首次运行向导启动问题故障排除 579
检查已停止的任务是否存在错误 579
服务事件消息 581
服务事件消息 582

指定的 CPU 或内存值无效 584
无法提取容器映像错误 585
CannotCreateContainerError: API error (500): devmapper 586
服务负载均衡器问题排查 587
启用 Docker 调试输出 588
Amazon ECS 日志文件位置 589
Amazon ECS 容器代理日志 589
Amazon ECS ecs-init 日志 589
任务凭证审核日志的 IAM 角色 590
Amazon ECS 日志收集器 590
代理自检诊断 591
Docker 诊断 593
列出 Docker 容器 593
查看 Docker 日志 594
检查 Docker 容器 594
API failures 错误消息 595
任务 IAM 角色问题排查 596
Windows 容器 599
Windows 容器注意事项 599
Windows 容器入门 600
步骤 1：创建 Windows 集群 600
步骤 2：在集群中启动 Windows 容器实例 600
步骤 3：注册 Windows 任务定义 603
步骤 4：使用您的任务定义创建服务 604
步骤 5：查看您的服务 604
Windows 任务定义 605
Windows 任务定义参数 605
Windows 示例任务定义 607
任务的 Windows IAM 角色 608
任务的 IAM 角色的容器引导脚本 608
将 Windows 映像推送到 Amazon ECR 608
文档历史记录 610
AWS 词汇表 623

什么是 Amazon Elastic Container Service？

Amazon Elastic Container Service (Amazon ECS) 是一项高度可扩展的快速容器管理服务，它可轻松运行、停止和管理集群上的 Docker 容器。您可以通过使用 Fargate 启动类型启动服务或任务，将集群托管在由 Amazon ECS 管理的无服务器基础设施上。若要进行更多控制，您可以在使用 EC2 启动类型进行管理的 Amazon Elastic Compute Cloud (Amazon EC2) 实例集群上托管您的任务。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。

利用 Amazon ECS，您可以通过简单的 API 调用来启动和停止基于容器的应用程序，可以从集中式服务获取集群状态，并且可以访问许多熟悉的 Amazon EC2 功能。

您可以根据资源需求、隔离策略和可用性要求使用 Amazon ECS 计划容器在集群中的放置。借助 Amazon ECS，无需操作自己的集群管理和配置管理系统，也无需担心扩展管理基础设施。

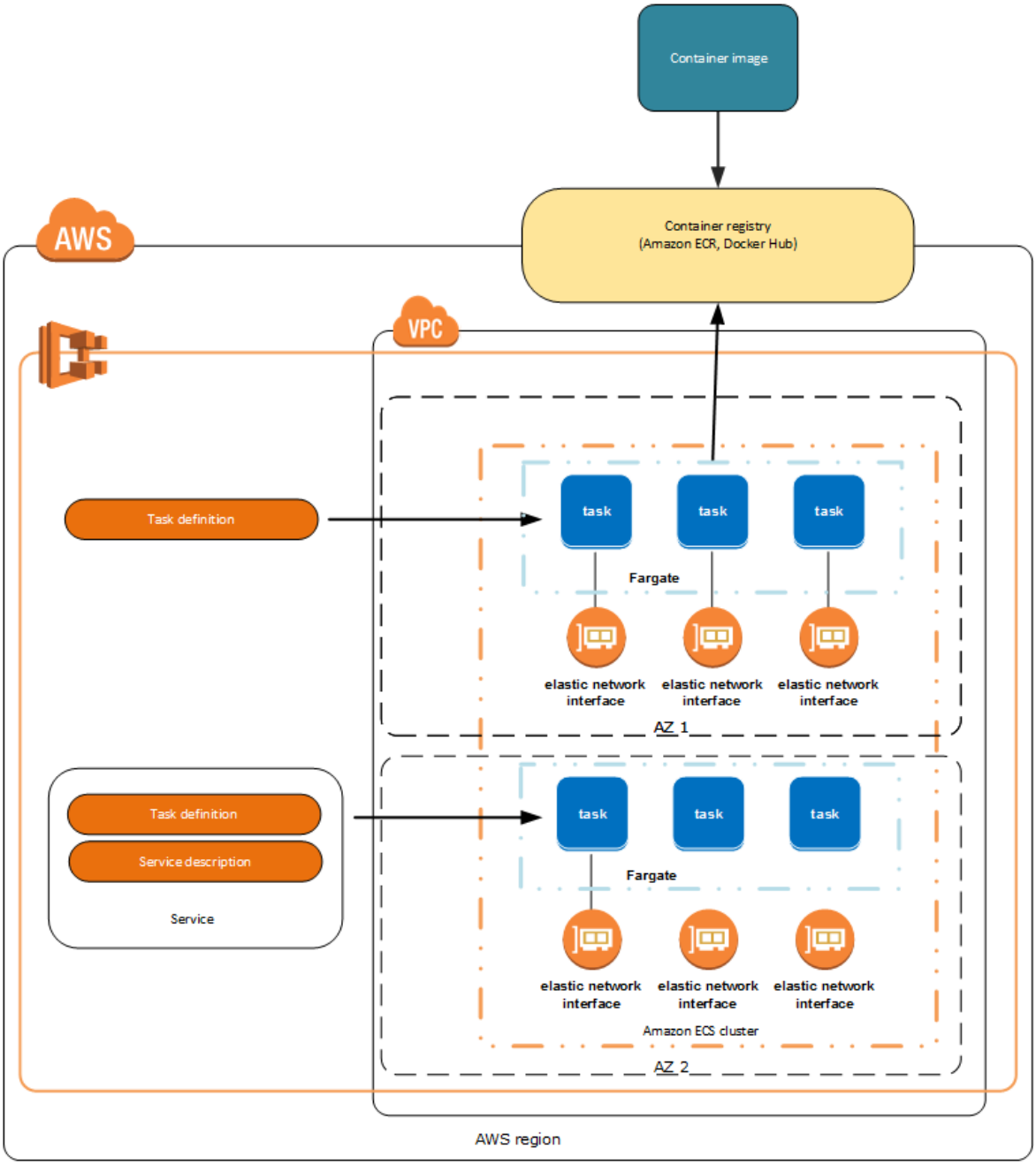
Amazon ECS 可用于创建一致的部署和构建体验、管理和扩展批处理和提取-转换-加载 (ETL) 工作负载以及在微服务模型上构建先进的应用程序架构。有关 Amazon ECS 使用案例和方案的更多信息，请参阅容器使用案例。

AWS Elastic Beanstalk 还可用于快速开发、测试和部署 Docker 容器以及应用程序基础设施的其他组件；而直接使用 Amazon ECS 将提供对一系列更广泛的使用案例的更精细的控制和访问。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南。

Amazon ECS 的功能

Amazon ECS 是一项区域服务，可在区域内的多个可用区中以高度可用的方式简化正在运行的应用程序容器。您可以在新的或现有的 VPC 中创建 Amazon ECS 集群。在集群启动并运行后，您可以定义用于指定在集群中运行的 Docker 容器映像的任务定义和服务。容器映像将在容器注册表中存储和提取，可能存在于您的 AWS 基础设施的内部或外部。

下图说明使用 Fargate 启动类型的 Amazon ECS 环境的架构：

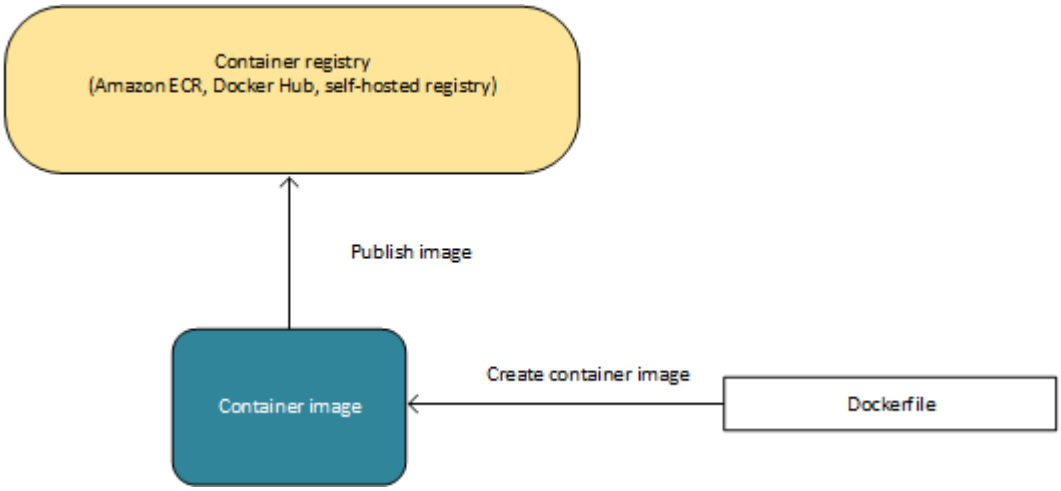


以下部分将更详细地深入探究 Amazon ECS 基础设施的各个元素。

容器和映像

要在 Amazon ECS 上部署应用程序，必须构建应用程序组件以在容器 中运行。Docker 容器是一个软件开发的标准化单位，包含您的软件应用程序需要运行的一切：代码、运行时、系统工具、系统库等。从称为映像的只读模板中创建容器。

映像通常通过 Dockerfile 进行构建，后者是一个纯文本文件，指定了容器中包含的所有组件。随后，这些映像将存储在注册表 中，可从该表中下载映像并在您的集群上运行映像。有关容器技术的更多信息，请参阅 Amazon ECS 的 Docker 基本知识 (p. 14)。



任务定义

为了准备要在 Amazon ECS 上运行的应用程序，需要创建任务定义。任务定义是一种 JSON 格式的文本文件，描述了构成应用程序的一个或多个容器 (最多为 10 个)。它可以被视为应用程序的蓝图。任务定义为您的应用程序指定各种参数。任务定义参数的示例有：要使用哪些容器，要使用哪种启动类型，应为应用程序打开哪些端口，应与任务中的容器一起使用哪些数据卷。可用于任务定义的特定参数取决于您使用的启动类型。有关创建任务定义的更多信息，请参阅 Amazon ECS 任务定义 (p. 43)。

以下是包含运行 NGINX Web 服务器的单个容器的任务定义的示例（使用 Fargate 启动类型）。有关演示在任务定义中使用多个容器的更扩展的示例，请参阅示例任务定义 (p. 112)。

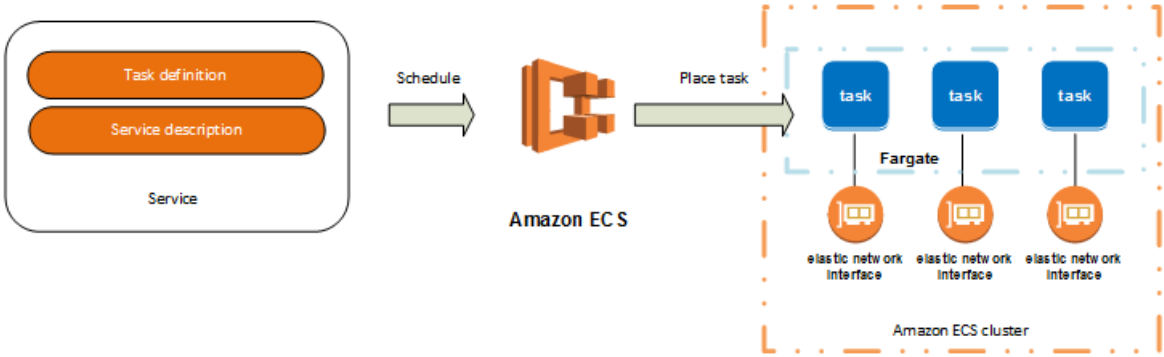
```
{
    "family": "webserver",
    "containerDefinitions": [
        {
            "name": "web",
            "image": "nginx",
            "memory": "100",
            "cpu": "99"
        },
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "memory": "512",
    "cpu": "256",
}
```

任务和计划

任务 是集群内的任务定义的实例化。在为 Amazon ECS 中的应用程序创建任务定义后，您可以指定将在集群上运行的任务的数量。

每个使用 Fargate 启动类型的任务都具有自己的隔离边界，不与其他任务共享底层内核、CPU 资源、内存资源或弹性网络接口。

Amazon ECS 任务计划程序负责将任务放置到您的集群中。有多种不同的计划选项可供选择。例如，您可以定义同时运行和维护指定数量的任务的服务。有关可用的不同计划选项的更多信息，请参阅计划 Amazon ECS 任务 (p. 250)。



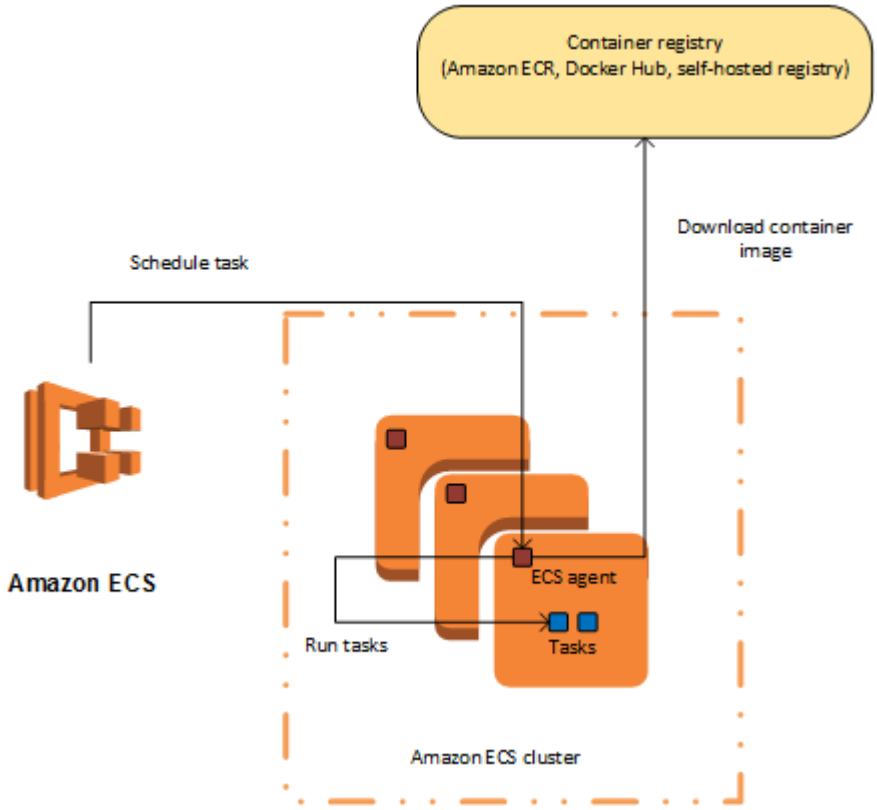
集群

使用 Amazon ECS 运行任务时，需要将任务放置在集群 上，这是资源的逻辑分组。当对集群中的任务使用 Fargate 启动类型时，Amazon ECS 将管理您的集群资源。在使用 EC2 启动类型时，您的集群是您管理的一组容器实例。Amazon ECS 容器实例是运行 Amazon ECS 容器代理的 Amazon EC2 实例。Amazon ECS 从您指定的注册表中下载容器映像，并在集群内运行这些映像。

有关创建集群的更多信息，请参阅 Amazon ECS 集群 (p. 38)。 如果使用的是 EC2 启动类型，可以在 Amazon ECS 容器实例 (p. 125)中阅读有关创建容器实例的信息。

容器代理

容器代理 在 Amazon ECS 集群内的每个基础设施资源上运行。它向 Amazon ECS 发送有关资源的当前正在运行的任务和资源使用率的信息，并且在接收来自 Amazon ECS 的请求时启动和停止任务。有关更多信息，请参阅Amazon ECS 容器代理 (p. 198)。



如何开始使用 Amazon ECS

如果是第一次使用 Amazon ECS，则 Amazon ECS 的 AWS 管理控制台将提供一个首次运行向导，该向导逐步指导您为 Web 服务器定义任务定义、配置服务以及启动您的首个 Fargate 任务。强烈建议之前从未使用过 Amazon ECS 的用户使用此首次运行向导。有关更多信息，请参阅 开始使用 Amazon ECS (p. 19)教程。

或者，您可以安装 AWS Command Line Interface (AWS CLI) 以使用 Amazon ECS。有关更多信息，请参阅Amazon ECS 的设置 (p. 7)。

相关服务

Amazon ECS 可与以下 AWS 服务一起使用：

AWS Identity and Access Management

IAM 是一项 Web 服务，可帮助您安全地控制用户对 AWS 资源的访问。通过 IAM 可以控制哪些人可以使用您的 AWS 资源（身份验证）以及他们可以使用的资源和采用的方式（授权）。在 Amazon ECS 中，IAM 可用于控制容器实例级别的访问（使用 IAM 角色）以及任务级别的访问（使用 IAM 任务角色）。有关更多信息，请参阅 适用于 Amazon Elastic Container Service 的 Identity and Access Management (p. 367)。

Amazon EC2 Auto Scaling

Auto Scaling 是一项 Web 服务，使您能够根据用户定义的策略、运行状况检查和时间表自动扩展或缩减任务。您可以对服务内的 Fargate 任务使用 Auto Scaling，从而根据多个指标进行扩展，或对 EC2 任务使用来扩展集群中的容器实例。有关更多信息，请参阅 服务 Auto Scaling (p. 299)。

Elastic Load Balancing

Elastic Load Balancing 在 Amazon ECS 服务中的任务间自动分配应用程序的传入流量。它可以让您实现更高水平的应用程序容错能力，同时持续提供分配应用程序流量所需的负载均衡容量。您可以使用 Elastic Load Balancing 创建一个使集群中的服务流量达到均衡的终端节点。有关更多信息，请参阅 服务负载均衡 (p. 284)。

Amazon Elastic Container Registry

Amazon ECR 是一项托管 AWS Docker 注册表服务，安全、可扩展且可靠。通过使用 IAM，Amazon ECR 支持具有基于资源的权限的私有 Docker 存储库，以便特定用户或任务可以访问存储库和映像。开发人员可以使用 Docker CLI 推送、拉取和管理映像。有关更多信息，请参见 Amazon Elastic Container Registry 用户指南。

AWS CloudFormation

AWS CloudFormation 向开发人员和系统管理员提供了一种简便地创建和管理一批相关的 AWS 资源的方法，并通过有序且可预测的方式对其进行预置和更新。您可以在 AWS CloudFormation 脚本中将集群、任务定义和服务定义为实体。有关更多信息，请参阅 AWS CloudFormation 模板参考。

访问 Amazon ECS

您可以通过以下方式使用 Amazon ECS：

AWS 管理控制台

此控制台是用于管理 Amazon ECS 资源的基于浏览器的界面。有关指导您使用控制台的教程，请参阅 开始使用 Amazon ECS (p. 19)。

AWS 命令行工具

您可以使用 AWS 命令行工具在系统的命令行上发出命令以执行 Amazon ECS 和 AWS 任务；这比使用控制台更快且更方便。命令行工具也非常适用于构建执行 AWS 任务的脚本。

AWS 提供两组命令行工具：AWS Command Line Interface (AWS CLI) 和 适用于 Windows PowerShell 的 AWS 工具。有关更多信息，请参阅 AWS Command Line Interface 用户指南和 适用于 Windows PowerShell 的 AWS 工具 用户指南。

Amazon ECS CLI

除了使用 AWS CLI 访问 Amazon ECS 资源外，您还可以使用 Amazon ECS CLI，它提供一些高级命令，可用于简化通过 Docker Compose 从本地开发环境创建、更新和监控集群和任务的过程。有关更多信息，请参阅使用 Amazon ECS 命令行界面 (p. 425)。

AWS 软件开发工具包

此外，我们还提供了软件开发工具包，这些软件开发工具包可让您通过一系列编程语言访问 Amazon ECS。软件开发工具包自动处理以下任务，如：

- 使用密码对服务请求签名
- 重试请求
- 处理错误响应

有关可用软件开发工具包的更多信息，请参阅适用于 Amazon Web Services 的工具。

Amazon ECS 的设置

如果您已注册 Amazon Web Services (AWS) 并已在使用 Amazon Elastic Compute Cloud (Amazon EC2)，您与使用 Amazon ECS 已近在咫尺。这两个服务的设置过程相似。以下指南将帮助您做好使用 Amazon ECS 首次运行向导或 Amazon ECS 命令行界面 (CLI) 启动首个集群的准备。

> **Note**
>
> 因为 Amazon ECS 使用 Amazon EC2 的许多组件，所以，您可以将 Amazon EC2 控制台用于这些步骤中的许多步骤。

要开始设置 Amazon ECS，请完成以下任务。如果您已完成以下任何步骤，可以将其跳过并继续安装自定义 AWS CLI。

注册 AWS

当您注册 AWS 时，您的 AWS 账户会自动注册所有服务，包括 Amazon EC2 和 Amazon ECS。您只需为使用的服务付费。

如果您已有 AWS 账户，请跳到下一个任务。如果您还没有 AWS 账户，请使用以下步骤创建。

创建 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

请记下您的 AWS 账号，因为在下一个任务中您会用到它。

创建 IAM 用户

AWS 中的服务（如 Amazon EC2 和 Amazon ECS）要求您在访问时提供凭证，以便服务可以确定您是否有权限访问其资源。控制台要求您的密码。您可以为您的 AWS 账户创建访问密钥以访问命令行界面或 API。但是，我们不建议您使用 AWS 账户的凭证访问 AWS，而建议您改用 AWS Identity and Access Management (IAM)。创建 IAM 用户，然后将该用户添加到具有管理权限的 IAM 组或授予此用户管理权限。然后，您就可以使用专门的 URL 和该 IAM 用户的凭证来访问 AWS。

如果您已注册 AWS 但尚未为自己创建一个 IAM 用户，则可以使用 IAM 控制台自行创建。

自行创建管理员用户并将该用户添加到管理员组（控制台）

1. 使用 AWS 账户电子邮件地址和密码，以 AWS 账户根用户 身份登录到 IAM 控制台 (https://console.aws.amazon.com/iam/)。

   > **Note**
   >
   > 强烈建议您遵守以下使用 **`Administrator`** IAM 用户的最佳实践，妥善保存根用户凭证。只在执行少数账户和服务管理任务时才作为根用户登录。

2. 在导航窗格中，选择用户，然后选择添加用户。

3. 对于 User name (用户名)，输入 **`Administrator`**。
4. 选中 AWS 管理控制台 访问 旁边的复选框。然后选择自定义密码，并在文本框中输入新密码。
5. （可选）默认情况下，AWS 要求新用户在首次登录时创建新密码。您可以清除 User must create a new password at next sign-in (用户必须在下次登录时创建新密码) 旁边的复选框以允许新用户在登录后重置其密码。
6. 选择下一步: 权限。
7. 在设置权限下，选择将用户添加到组。
8. 选择创建组。
9. 在 Create group (创建组) 对话框中，对于 Group name (组名称)，输入 **`Administrators`**。
10. 选择 Filter policies (筛选策略)，然后选择 AWS managed-job function (AWS 托管的工作职能) 以筛选表内容。
11. 在策略列表中，选中 AdministratorAccess 的复选框。然后选择 Create group (创建组)。

    Note

    您必须先激活 IAM 用户和角色对账单的访问权限，然后才能使用 `AdministratorAccess` 权限访问 AWS Billing and Cost Management 控制台。为此，请按照"向账单控制台委派访问权限"教程第 1 步中的说明进行操作。
12. 返回到组列表中，选中您的新组所对应的复选框。如有必要，选择 Refresh 以在列表中查看该组。
13. 选择下一步: 标签。
14. （可选）通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南 中的标记 IAM 实体。
15. 选择 Next: Review (下一步: 审核) 以查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择 Create user。

您可使用此相同的流程创建更多的组和用户，并允许您的用户访问 AWS 账户资源。要了解有关使用策略限制用户对特定 AWS 资源的权限的信息，请参阅访问管理和示例策略。

要以该新 IAM 用户的身份登录，请从 AWS 控制台注销，然后使用以下 URL，其中 your_aws_account_id 是您不带连字符的 AWS 账号（例如，如果您的 AWS 账号是 `1234-5678-9012`，则您的 AWS 账户 ID 是 `123456789012`）：

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

输入您刚创建的 IAM 用户名和密码。登录后，导航栏显示 your_user_name @ your_aws_account_id。

如果您不希望您的登录页面 URL 包含 AWS 账户 ID，可以创建账户别名。从 IAM 控制面板中，选择 Create Account Alias (创建账户别名)，然后输入一个别名，例如您的公司名称。要在创建账户别名后登录，请使用以下 URL：

```
https://your_account_alias.signin.aws.amazon.com/console/
```

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 users sign-in link (IAM 用户登录链接) 下进行检查。

有关 IAM 的更多信息，请参阅 AWS Identity and Access Management 用户指南。

创建 IAM 角色

在 Amazon ECS 容器代理可以代表您调用 Amazon ECS API 操作之前，它需要服务的 IAM 策略和角色，以便了解属于您的代理。

对于使用 EC2 启动类型的任务，您可以创建一个 IAM 角色，该角色可让代理知道应向哪个账户注册容器实例。当您使用 Amazon 通过此角色提供的经 Amazon ECS 优化的 AMI 启动容器实例时，代理会自动将容器实例注册到 `default` 集群中。此角色称为 Amazon ECS 容器实例 IAM 角色。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

Amazon ECS 容器代理还会代表您调用 Amazon EC2 和 Elastic Load Balancing API，以便在负载均衡器中注册和取消注册容器实例。您必须在启动服务前为其创建一个 IAM 角色，然后才能将负载均衡器附加到 Amazon ECS 服务。此要求适用于您计划用于负载均衡器的任意 Amazon ECS 服务。此角色称为 Amazon ECS 服务计划程序 IAM 角色。有关更多信息，请参阅 Amazon ECS 服务计划程序 IAM 角色 (p. 410)。

对于使用 Fargate 启动类型的任务，您可以创建一个 IAM 角色，该角色允许代理从 Amazon ECR 中提取容器映像或者使用 awslogs 日志驱动程序，该驱动程序是当前唯一受支持的此启动类型的日志记录选项。此角色称为 Amazon ECS 任务执行 IAM 角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色 (p. 406)。

> **Note**
>
> 在 Amazon ECS 控制台首次运行体验中，将自动为您创建这些 IAM 角色，因此，如果您打算使用控制台，则可以继续下一个部分。如果您不打算使用控制台，而是计划使用 AWS CLI，则需要手动创建这些 IAM 角色。

创建密钥对

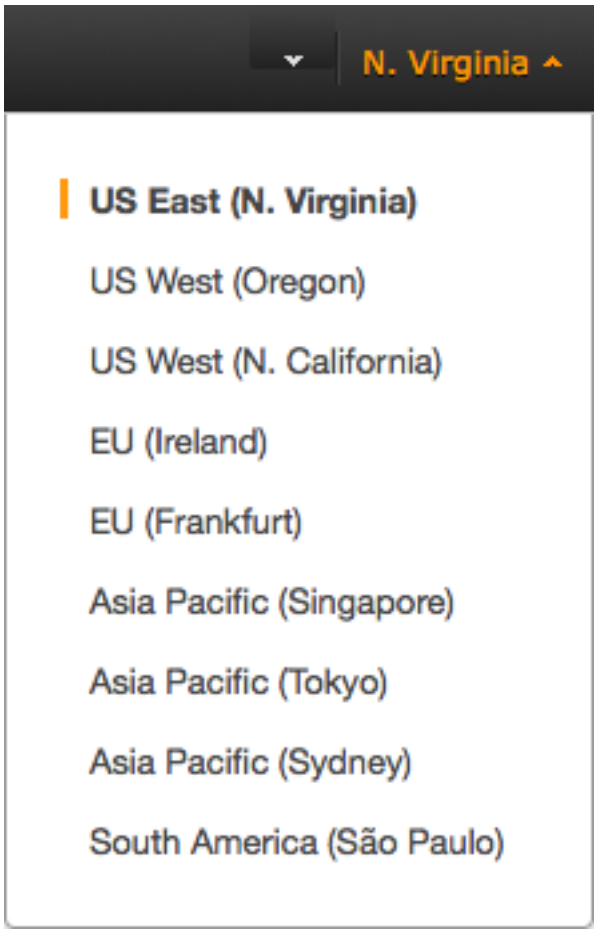
对于 Amazon ECS，密钥对只有在您打算使用 EC2 启动类型时才需要。

AWS 使用公有密钥密码术来保护实例的登录信息。 Linux 实例（例如 Amazon ECS 容器实例）没有用于 SSH 访问的密码。您使用密钥对安全地登录到实例。您可以在启动容器实例时指定密钥对的名称，然后提供私有密钥（使用 SSH 登录时）。

如果您尚未创建密钥对，则可以通过 Amazon EC2 控制台自行创建。如果您计划在多个区域启动实例，则需要在每个区域中创建密钥对。有关区域的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的区域和可用区。

创建密钥对

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中，选择密钥对所在的区域。您可以选择向您提供的任何区域，无需理会您身处的位置。但是，密钥对特定于区域。例如，如果您计划在美国东部（俄亥俄）区域中启动容器实例，则必须在美国东部（俄亥俄）区域中为实例创建密钥对。



3. 在导航窗格中的 NETWORK & SECURITY 下，选择 Key Pairs。

   Tip

   导航窗格位于控制台的左侧。如果您看不到窗格，它可能被最小化了；请选择箭头展开该窗格。您可能必须向下滚动才能看到 Key Pairs 链接。

4. 选择 Create Key Pair。
5. 在 Create Key Pair 对话框的 Key pair name 字段中输入新密钥对的名称，然后选择 Create。使用一个容易记住的名称 (如您的 IAM 用户名) 后跟 `-key-pair` 加区域名称。例如，me-key-pair-useast2。
6. 您的浏览器会自动下载私有密钥文件。基本文件名是您为密钥对指定的名称，文件扩展名为 `.pem`。将私有密钥文件保存在安全位置。

> **Important**
>
> 这是您保存私有密钥文件的唯一机会。启动实例时，提供密钥对的名称；每次连接到实例时，提供相应的私有密钥。

7. 如果您在 macOS 或 Linux 计算机上使用 SSH 客户端连接到您的 Linux 实例，请使用以下命令设置您私有密钥文件的权限，以确保只有您可以读取它。

```
chmod 400 your_user_name-key-pair-region_name.pem
```

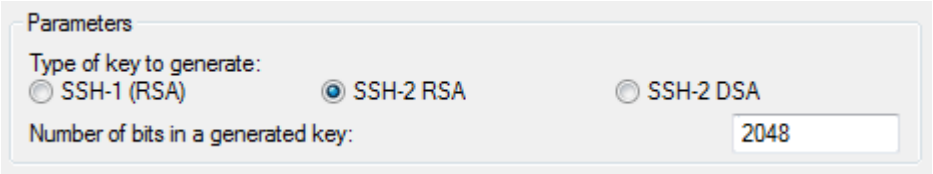
有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 密钥对 。

使用密钥对连接到实例

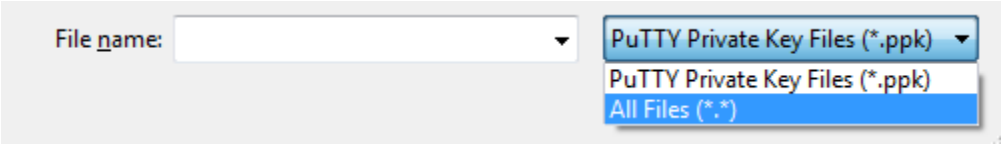
要从运行 macOS 或 Linux 的计算机连接到 Linux 实例，需要使用 `-i` 选项对 SSH 客户端指定 `.pem` 文件和私有密钥的路径。若要从运行 Windows 的计算机连接到 Linux 实例，可以使用 MindTerm 或 PuTTY。如果您计划使用 PuTTY，则需要安装它并遵循以下过程将 `.pem` 文件转换为 `.ppk` 文件。

（可选）准备使用 PuTTY 从 Windows 连接到 Linux 实例

1. 从 http://www.chiark.greenend.org.uk/~sgtatham/putty/ 下载并安装 PuTTY。请务必安装整个套件。
2. 启动 PuTTYgen（例如，在开始菜单中，依次单击所有程序 > PuTTY > PuTTYgen ）。
3. 在 Type of key to generate 下，选择 RSA。



4. 选择 Load。默认情况下，PuTTYgen 仅显示扩展名为 `.ppk` 的文件。要找到您的 `.pem` 文件，请选择显示所有类型的文件的选项。



5. 选择您在上一个过程中创建的私有密钥文件，然后选择 Open。选择 OK 关闭确认对话框。
6. 选择 Save private key (保存私有密钥)。PuTTYgen 会显示一条警告，提示将在未提供口令的情况下保存密钥。选择是。
7. 为密钥指定密钥对所用的相同名称。PuTTY 会自动添加 `.ppk` 文件扩展名。

创建 Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) 允许您在已定义的虚拟网络内启动 AWS 资源。强烈建议您在 VPC 中启动您的容器实例。

> **Note**
>
> Amazon ECS 控制台首次运行体验会为您的集群创建 VPC，因此，如果您打算使用 Amazon ECS 控制台，则可以跳到下一个部分。

如果您有默认 VPC，也可以跳过此部分并进入下一个任务，即 创建安全组 (p. 12)。要确定您是否具有默认 VPC，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 控制台中支持的平台。否则，您可以使用以下步骤在账户中创建非默认 VPC。

Important

如果您的账户在某个区域中支持 Amazon EC2 Classic，则您在该区域没有默认 VPC。

创建非默认 VPC

1. 打开 Amazon VPC 控制台 https://console.aws.amazon.com/vpc/。
2. 从导航栏中，为 VPC 选择区域。VPC 特定于某一区域，因此您应选择已创建密钥对的区域。
3. 在 VPC 控制面板上，选择 Launch VPC Wizard (启动 VPC 向导)。
4. 在 Step 1: Select a VPC Configuration 页面上，确保选中 VPC with a Single Public Subnet，然后选择 Select。
5. 在 Step 2: VPC with a Single Public Subnet (步骤 2: 带有单个公有子网的 VPC) 页面上，在 VPC name (VPC 名称) 字段中为您的 VPC 输入友好名称。保留其他默认配置设置，然后选择 Create VPC。在确认页面上，请选择 OK。

有关 Amazon VPC 的更多信息，请参阅 Amazon VPC 用户指南 中的 Amazon VPC 是什么？。

创建安全组

安全组用作相关容器实例的防火墙，可在容器实例级别控制入站和出站流量。您可以向安全组添加规则，以便使用 SSH 从您的 IP 地址连接到容器实例。您还可以添加允许来自任意位置的入站和出站 HTTP 和 HTTPS 访问的规则。向任务所需的开放端口添加任意规则。容器实例需要外部网络访问来与 Amazon ECS 服务终端节点通信。

Note

Amazon ECS 控制台首次运行时会为您的实例创建安全组并根据您使用的任务定义创建负载均衡器，因此，如果要使用 Amazon ECS 控制台，则可以跳到下一个部分。

如果您计划在多个区域中启动容器实例，则需要在每个区域中创建安全组。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例） 中的区域和可用区。

Tip

您需要本地计算机的公有 IP 地址，可以使用服务获得该地址。例如，我们提供以下服务：http://checkip.amazonaws.com/ 或 https://checkip.amazonaws.com/。要查找另一项可提供您的 IP 地址的服务，请使用搜索短语“what is my IP address”。 如果您通过 Internet 服务提供商 (ISP) 连接或者在不使用静态 IP 地址的情况下从防火墙后面连接，则必须找出客户端计算机使用的 IP 地址范围。

创建具有最小特权的安全组

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中选择安全组的区域。安全组特定于某一区域，因此您应选择已创建密钥对的区域。
3. 在导航窗格中，选择 Security Groups，然后选择 Create Security Group。
4. 输入新安全组的名称和描述。选择一个您容易记住的名称，例如 ecs-instances-default-cluster。
5. 在 VPC 列表中，确保选择了您的默认 VPC。它标有星号 (*)。

   Note

   如果您的账户支持 Amazon EC2 Classic，请选择您在上一个任务中创建的 VPC。
6. Amazon ECS 容器实例不需要打开任何入站端口。但您可能需要添加 SSH 规则，以便登录容器实例并使用 Docker 命令检查任务。如果您希望容器实例托管运行 Web 服务器的任务，也可以添加适用于 HTTP 和 HTTPS 的规则。容器实例需要外部网络访问来与 Amazon ECS 服务终端节点通信。完成以下步骤可添加这些可选的安全组规则。

在 Inbound 选项卡上，创建以下规则 (为每个新规则选择 Add Rule)，然后选择 Create：

- 从 Type 列表中选择 HTTP，确保 Source 设置为 Anywhere (`0.0.0.0/0`)。
- 从 Type 列表中选择 HTTPS，确保 Source 设置为 Anywhere (`0.0.0.0/0`)。
- 从 Type 列表中选择 SSH。在 Source 字段中，确保选中 Custom IP，然后采用 CIDR 表示法指定您计算机或网络的公有 IP 地址。要采用 CIDR 表示法指定单个 IP 地址，请添加路由前缀 `/32`。例如，如果您的 IP 地址是 `203.0.113.25`，请指定 `203.0.113.25/32`。如果您的公司要分配同一范围内的地址，请指定整个范围，例如 `203.0.113.0/24`。

  Important

  出于安全原因，我们不建议您允许从所有 IP 地址 (`0.0.0.0/0`) 对您的实例进行 SSH 访问（以测试为目的的短暂访问除外）。

安装 AWS CLI

可以在 AWS 管理控制台中手动管理 Amazon ECS 的所有操作。但是，您可以在本地桌面上安装 AWS CLI 或开发人员工具包并生成脚本，以便在 Amazon ECS 中自动执行常见管理任务。

要对 Amazon ECS 使用 AWS CLI，请安装最新版本的 AWS CLI。有关安装 AWS CLI 或升级到最新版本的信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

Amazon ECS 的 Docker 基本知识

Docker 是一项可让您构建、运行、测试和部署基于 Linux 容器的分布式应用程序的技术。Amazon ECS 使用任务定义中的 Docker 映像来启动集群中的 Amazon EC2 实例上的容器。有关 Amazon ECS 产品详细信息、特色客户案例研究和常见问题，请参阅 Amazon Elastic Container Service 产品详细信息页面。

本指南中的文档假定读者已基本了解 Docker 是什么及其工作方式。有关 Docker 的更多信息，请参阅 Docker 是什么？和 Docker 概述。

主题

- 安装 Docker (p. 14)
- 创建 Docker 映像 (p. 15)
- (可选) 将您的映像推送至 Amazon Elastic Container Registry (p. 16)
- （可选）清除 (p. 17)
- 后续步骤 (p. 17)

安装 Docker

> **Note**
>
> 如果您已安装 Docker，请跳到创建 Docker 映像 (p. 15)。

Docker 适用于许多不同的操作系统，包括大多数现代 Linux 分发版 (如 Ubuntu) 甚至 Mac OSX 和 Windows。有关如何在特定的操作系统上安装 Docker 的更多信息，请转到 Docker 安装指南。

您甚至无需本地开发系统即可使用 Docker。如果您已使用 Amazon EC2，则可启动一个实例并安装 Docker 以开始使用。

在 Amazon EC2 实例上安装 Docker

1. 使用 Amazon Linux 2 AMI 启动实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的启动实例。
2. 连接到您的实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的连接到您的 Linux 实例。
3. 更新实例上已安装的程序包和程序包缓存。

```
sudo yum update -y
```

4. 安装最新的 Docker Community Edition 程序包。

```
sudo amazon-linux-extras install docker
```

5. 启动 Docker 服务。

```
sudo service docker start
```

6. 将 `ec2-user` 添加到 `docker` 组，以便您能够执行 Docker 命令，而无需使用 `sudo`。

```
sudo usermod -a -G docker ec2-user
```

7. 退出，再重新登录以接受新的 `docker` 组权限。您可以关闭当前的 SSH 终端窗口并在新终端窗口中重新连接到实例，完成这一过程。您的新 SSH 会话将具有相应的 `docker` 组权限。

8. 验证 `ec2-user` 是否能在没有 `sudo` 的情况下运行 Docker 命令。

```
docker info
```

Note

在某些情况下，您可能需要重新启动实例，以便为 `ec2-user` 提供访问 Docker 守护程序的权限。如果您看到以下错误，请尝试重启您的实例：

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

创建 Docker 映像

Amazon ECS 任务定义使用 Docker 映像启动集群中的容器实例上的容器。在本节中，您将创建简单 Web 应用程序的 Docker 映像，并在本地系统或 EC2 实例上测试此映像，然后将此映像推送至容器注册表（如 Amazon ECR 或 Docker Hub），以便能够在 ECS 任务定义中使用它。

创建简单 Web 应用程序的 Docker 映像

1. 创建名为 `Dockerfile` 的文件。Dockerfile 是一个清单文件，描述了用于 Docker 映像的基本映像以及要安装的项目以及在此项目上运行的内容。有关 Dockerfile 的更多信息，请转到 Dockerfile 参考。

```
touch Dockerfile
```

2. 编辑您刚刚创建的 `Dockerfile` 并添加以下内容。

```
FROM ubuntu:16.04

# Install dependencies
RUN apt-get update
RUN apt-get -y install apache2

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh
RUN echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh
RUN echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh
RUN echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh
RUN chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

此 Dockerfile 使用 Ubuntu 16.04 映像。`RUN` 指令将更新程序包缓存，为 Web 服务器安装一些软件包，然后将“Hello World!”内容添加到 Web 服务器的文档根目录。`EXPOSE` 指令在容器上公开端口 80，`CMD` 指令启动 Web 服务器。

3. 从您的 Dockerfile 生成 Docker 映像。

Note

Docker 的某些版本可能需要在以下命令中使用 Dockerfile 完整路径，而不是所示的相对路径。

```
docker build -t hello-world .
```

4. 运行 docker images 以验证是否已正确创建映像。

```
docker images --filter reference=hello-world
```

输出:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago       241MB
```

5. 运行新构建的映像。`-p 80:80` 选项将容器上公开的端口 80 映射到主机系统上的端口 80。有关 docker run 的更多信息，请转到 Docker 运行参考。

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> 来自 Apache Web 服务器的输出将显示在终端窗口中。您可以忽略“`Could not reliably determine the server's fully qualified domain name`”消息。

6. 打开浏览器并指向正在运行 Docker 并托管您的容器的服务器。

   - 如果您使用的是 EC2 实例，这将是服务器的 Public DNS 值，此值与您用于通过 SSH 连接到实例的地址相同。确保实例的安全组允许端口 80 上的入站流量。
   - 如果您正在本地运行 Docker，可将您的浏览器指向 http://localhost/。
   - 如果您正在 Windows 或 Mac 计算机上使用 docker-machine，请使用 docker-machine ip 命令查找托管 Docker 的 VirtualBox VM 的 IP 地址，并将 *`machine-name`* 替换为您正在使用的 Docker 计算机的名称。

   ```
   docker-machine ip machine-name
   ```

   您应看到一个显示“Hello World!”语句的网页。

7. 通过键入 Ctrl + c 来停止 Docker 容器。

(可选) 将您的映像推送至 Amazon Elastic Container Registry

Amazon ECR 是一项托管 AWS Docker 注册表服务。客户可以使用熟悉的 Docker CLI 推送、拉取和管理映像。有关 Amazon ECR 产品详细信息、特色客户案例研究和常见问题，请参阅 Amazon Elastic Container Registry 产品详细信息页面。

本节要求以下内容：

- 您已经安装并配置 AWS CLI。如果您尚未在系统中安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS Command Line Interface。
- 您的用户具有访问 Amazon ECR 服务所需的 IAM 权限。有关更多信息，请参阅 Amazon ECR 托管策略。

标记映像并将其推送至 Amazon ECR

1. 创建用于存储您的 `hello-world` 映像的 Amazon ECR 存储库。在输出中记下 `repositoryUri`。

```
aws ecr create-repository --repository-name hello-repository --region region
```

输出:

```
{
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "hello-repository",
        "repositoryArn": "arn:aws:ecr:region:aws_account_id:repository/hello-
repository",
        "createdAt": 1505337806.0,
        "repositoryUri": "aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository"
    }
}
```

2. 使用上一步中的 `repositoryUri` 值标记 `hello-world` 映像。

```
docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

3. 运行 aws ecr get-login --no-include-email 命令以获取您的注册表的 docker login 身份验证命令字符串。

   Note

   AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本（17.06 或更高版本），我们建议使用 1.11.91 版或更高版本。可以使用 aws --version 命令查看 AWS CLI 版本。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的 安装 AWS 命令行界面。

```
aws ecr get-login --no-include-email --region region
```

4. 运行上一步中返回的 docker login 命令。此命令提供一个在 12 小时内有效的授权令牌。

   Important

   在您执行此 docker login 命令时，系统上的其他用户可以在进程列表 (ps -e) 显示中看到该命令字符串。由于 docker login 命令包含身份验证凭证，因此系统上的其他用户可按此方式查看凭证会带来风险。他们可能会使用凭证获取对您的存储库的推送和拉取访问权限。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p password` 选项并在系统提示时输入密码来以交互方式登录。

5. 使用上一步中的 `repositoryUri` 值将映像推送至 Amazon ECR。

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

（可选）清除

完成试验 Amazon ECR 映像后，您可以删除存储库，从而无需为映像存储付费。

```
aws ecr delete-repository --repository-name hello-repository --region region --force
```

后续步骤

现在，您已经创建了一个 Docker 映像并将其推送到 Amazon ECR 存储库，您可以开始创建您的 Amazon ECS 资源以启动容器了。使用以下主题继续：

- 完成先决条件。有关更多信息，请参阅 Amazon ECS 的设置 (p. 7)。
- 有关 AWS CLI 演练的更多信息，请参阅 将 AWS CLI 与 Amazon ECS 结合使用 (p. 525)。
- 有关 AWS 管理控制台演练的更多信息，请参阅 开始使用 Amazon ECS (p. 19)。

开始使用 Amazon ECS

通过启动您的第一个任务所需的 Amazon ECS 资源，开始使用 Amazon Elastic Container Service (Amazon ECS) 。Amazon ECS 控制台提供了简化这一过程的首次运行体验。

在支持 AWS Fargate 的区域中，Amazon ECS 首次运行向导将引导您完成通过 Fargate 开始使用 Amazon ECS 的过程。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。该向导提供了创建集群和启动示例 Web 应用程序的选项。如果您已有在 Amazon ECS 中启动的 Docker 映像，可以使用该映像创建任务定义并改为将其用于集群。

在不支持 AWS Fargate 的区域中，Amazon ECS 首次运行向导将引导您完成通过 EC2 开始使用任务的过程。该向导提供了创建集群和启动示例 Web 应用程序的选项。如果您已有在 Amazon ECS 中启动的 Docker 映像，可以使用该映像创建任务定义并改为将其用于集群。

主题

- 通过 Fargate 开始使用 Amazon ECS (p. 19)
- 开始使用 Amazon ECS (p. 22)
- 清除 Amazon ECS 资源 (p. 26)

通过 Fargate 开始使用 Amazon ECS

通过在 Amazon ECS 控制台中创建使用 Fargate 运行类型的任务定义，计划任务和配置集群，开始使用 Amazon Elastic Container Service (Amazon ECS)。

在支持 AWS Fargate 的区域中，Amazon ECS 首次运行向导将引导您完成通过 Fargate 开始使用 Amazon ECS 的过程。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。该向导提供了创建集群和启动示例 Web 应用程序的选项。如果您已有在 Amazon ECS 中启动的 Docker 映像，可以使用该映像创建任务定义并改为将其用于集群。

> **Important**
>
> 有关 EC2 任务的 Amazon ECS 首次运行向导的更多信息，请参阅 Amazon ECS 入门。

完成下面的任务，开始通过 Fargate 使用 Amazon ECS：

先决条件

在开始之前，请确保完成Amazon ECS 的设置 (p. 7) 中的步骤，并且您的 AWS 用户具有 `AdministratorAccess` 或 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的权限。

首次运行向导会尝试自动创建任务执行 IAM 角色，这是 Fargate 任务需要的。要确保首次运行时能够创建该 IAM 角色，必须满足以下条件之一：

- 您的用户拥有管理员权限。有关更多信息，请参阅 Amazon ECS 的设置 (p. 7)。
- 您的用户拥有创建服务角色的 IAM 权限。有关更多信息，请参阅创建角色以向 AWS 服务委派权限。
- 拥有管理员权限的用户手动创建了任务执行角色，使之对要所用的账户可用。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

步骤 1：创建任务定义

任务定义类似于应用程序的蓝图。每次在 Amazon ECS 中启动任务时，您都指定任务定义。这样，服务知道要用于容器的 Docker 映像、任务中要使用的容器数量以及为每个容器分配的资源。

1. 打开 https://console.aws.amazon.com/ecs/home#/firstRun 上的 Amazon ECS 控制台首次运行向导。
2. 从导航栏中，选择美国东部(弗吉尼亚北部) 区域。

   Note

   您可以使用这些步骤对支持 Amazon ECS 使用 Fargate 的任何区域完成此首次运行向导。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。
3. 配置容器定义参数。

   对于 Container definition (容器定义)，首次运行向导会预加载控制台中的 `sample-app`、`nginx` 和 `tomcat-webserver` 容器定义。您可以选择重命名容器，或者通过选择 Edit 并编辑显示的值来查看和编辑容器使用的资源 (例如 CPU 单位和内存限制)。有关更多信息，请参阅容器定义 (p. 53)。

   Note

   如果在容器定义中使用 Amazon ECR 映像，请确保对 Amazon ECR 映像使用完整的 `registry/repository:tag` 命名。例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*。
4. 对于 Task definition (任务定义)，首次运行向导将定义任务定义，以与预加载的容器定义结合使用。您可以选择重命名任务定义，或者通过选择 Edit 并编辑显示的值来编辑任务使用的资源 (例如 Task memory 和 Task CPU 值)。有关更多信息，请参阅任务定义参数 (p. 52)。

   为简单起见，首次运行向导中创建的任务定义仅限用于一个容器。您之后可在 Amazon ECS 控制台中创建多容器任务定义。
5. 选择 Next (下一步)。

步骤 2：配置服务

在向导的这一部分中，可选择通过任务定义创建的 Amazon ECS 服务的配置方式。一个服务将会启动并在您的集群中维护指定数量的任务定义副本。Amazon ECS sample 应用程序是一个基于 Web 的 Hello World–样式应用程序，可无限期地运行。通过将它作为服务运行，它在任务状况不正常或意外停止时会重新启动。

首次运行向导会预加载一个服务定义，您可以看到在控制台中定义的 `sample-app-service` 服务。您可以选择重命名服务，或者通过选择 Edit 并执行以下操作来查看和编辑详细信息：

1. 在 Service name 字段中，为您的服务选择一个名称。
2. 在 Number of desired tasks (所需任务的数量) 字段中，输入要使用指定的任务定义启动的任务的数量。
3. 在 Security Group (安全组) 字段中，以 CIDR 块表示法指定 IPv4 地址范围，以允许入站流量。例如：`203.0.113.0/24`。
4. （可选）您可选择对服务使用 应用程序负载均衡器 负载均衡器。如果从配置为使用负载均衡器的服务启动任务，会向负载均衡器注册任务。来自负载均衡器的流量分布到负载均衡器中的各实例上。有关更多信息，请参阅 Application Load Balancer 简介。

   Important

   AWS 资源中有 应用程序负载均衡器 时，将产生相应费用。有关更多信息，请参阅 应用程序负载均衡器 定价。

   完成下列步骤以对服务使用负载均衡器。

- 在 Container to load balance (用于负载均衡的容器) 部分中，选择 Load balancer listener port (负载均衡器侦听端口)。此处的默认值是为示例应用程序设置的，但您可为负载均衡器配置不同的侦听器选项。有关更多信息，请参阅服务负载均衡 (p. 284)。

5. 查看服务设置，然后依次单击 Save、Next。

步骤 3：配置集群

在向导的这一部分，需要为集群命名，然后 Amazon ECS 为您进行联网和 IAM 配置。

1. 在 Cluster name 字段中，选择集群的名称。
2. 单击 Next 继续。

步骤 4: 检查

1. 查看您的任务定义、任务配置和集群配置，然后单击 Create (创建) 完成操作。您会定向到 Launch Status (启动状态) 页面，此页面显示您的启动状态。它描述此过程的每一步（在创建 Auto Scaling 组时，这可能需要几分钟才能完成和填充）。
2. 在启动完成后，选择 View service (查看服务)。

步骤 5：（可选）查看您的服务

如果您的服务是基于 Web 的应用程序 (如 Amazon ECS sample 应用程序)，可以使用 Web 浏览器查看其容器。

1. 在 Service: ***service-name*** 页面上，选择 Tasks 选项卡。
2. 从您的服务中的任务列表中选择任务。
3. 在 Network 部分中，选择您的任务的 ENI Id。这将转到 Amazon EC2 控制台，您可以在其中查看与任务关联的网络接口的详细信息，包括 IPv4 Public IP (IPv4 公有 IP) 地址。
4. 在 Web 浏览器中，输入 IPv4 Public IP (IPv4 公有 IP) 地址，您应该可以看到显示 Amazon ECS sample 应用程序的网页。

开始使用 Amazon ECS

通过在 Amazon ECS 控制台中创建使用 EC2 运行类型的任务定义，计划任务和配置集群，开始使用 Amazon Elastic Container Service (Amazon ECS)。有关更多信息，请参阅Amazon ECS 启动类型 (p. 81)。

在不支持 AWS Fargate 的区域中，Amazon ECS 首次运行向导将引导您完成通过 EC2 开始使用任务的过程。该向导提供了创建集群和启动示例 Web 应用程序的选项。如果您已有在 Amazon ECS 中启动的 Docker 映像，可以使用该映像创建任务定义并改为将其用于集群。

> Important
>
> 有关 Amazon ECS首次运行向导中 Fargate 任务的信息，请参阅 开始使用 Amazon ECS (p. 19)。

您也可选择创建 Amazon Elastic Container Registry (Amazon ECR) 映像存储库并将映像推送到库中。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

完成以下任务，开始使用 Amazon ECS：

先决条件

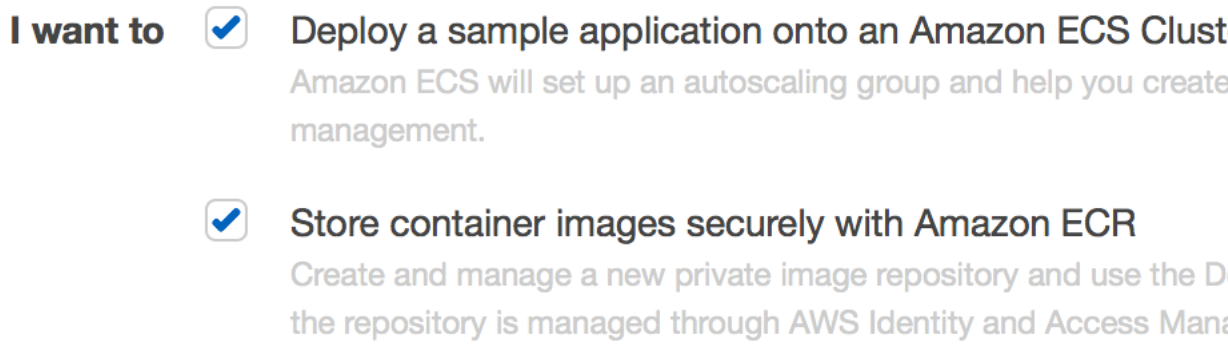
在开始之前，请确保完成Amazon ECS 的设置 (p. 7) 中的步骤，并且您的 AWS 用户具有 `AdministratorAccess` 或 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的权限。

首次运行向导会尝试自动创建 Amazon ECS 服务 IAM 和容器实例 IAM 角色。要确保首次运行时能够创建这些 IAM 角色，必须满足以下条件之一：

- 您的用户拥有管理员权限。有关更多信息，请参阅 Amazon ECS 的设置 (p. 7)。
- 您的用户拥有创建服务角色的 IAM 权限。有关更多信息，请参阅创建角色以向 AWS 服务委派权限。
- 一个拥有管理员权限的用户已手动创建这些 IAM 角色，使之对所用的账户可用。有关更多信息，请参阅 Amazon ECS 服务计划程序 IAM 角色 (p. 410) 和 Amazon ECS 容器实例 IAM 角色 (p. 404)。

步骤 1：选择您的配置选项

1. 打开 https://console.aws.amazon.com/ecs/home#/firstRun 上的 Amazon ECS 控制台首次运行向导。
2. 选择您的 Amazon ECS 首次运行选项。



要创建 Amazon ECS 集群并向其部署容器应用程序，请选中顶部选项。要创建 Amazon ECR 存储库并将映像推送到库中（您可在 Amazon ECS 任务定义中使用该库），请选中底部选项。选择 Continue (继续)。

> **Important**
>
> 如果您的范围限定在支持 AWS Fargate 的区域中，则不会看到这些选项。有关更多信息，请参阅 通过 Fargate 开始使用 Amazon ECS (p. 19)。

3. 如果您选择创建 Amazon ECR 存储库，则完成首次运行向导接下来的两个部分 Configure repository (配置存储库) 和 Build, tag, and push Docker image (构建、标记和推送 Docker 映像)。如果不创建 Amazon ECR 存储库，则向前跳至步骤 2：创建任务定义 (p. 23)。

配置存储库

存储库是您在 Amazon ECR 中存储 Docker 映像的位置。当您在 Amazon ECR 中推送或拉取映像时，您将指定注册表和存储库位置以告知 Docker 将映像推送到哪个位置或从哪个位置拉取映像。

1. 选择 Get Started 。
2. 对于 Repository configuration (存储库配置)，输入存储库的唯一名称并选择 Create repository (创建存储库)。

构建、标记和推送 Docker 映像

在向导的此部分中，您使用 Docker CLI 标记现有本地映像 (您从 Dockerfile 构建或从另一个注册表中拉取的映像，例如 Docker Hub)，然后将标记的映像推送到 Amazon ECR 注册表。

1. 检索 docker login 命令，此命令可用于通过将控制台中的 aws ecr get-login 命令粘贴到终端窗口中来对注册表验证 Docker 客户端的身份。

   > **Note**
   >
   > AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本（17.06 或更高版本），我们建议使用 1.11.91 版或更高版本。可以使用 aws --version 命令查看 AWS CLI 版本。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的 安装 AWS 命令行界面。

2. 运行上一步中返回的 docker login 命令。此命令提供一个在 12 小时内有效的授权令牌。

   > **Important**
   >
   > 在您执行此 docker login 命令时，系统上的其他用户可以在进程列表 (ps -e) 显示中看到该命令字符串。由于 docker login 命令包含身份验证凭证，因此系统上的其他用户可按此方式查看凭证会带来风险。他们可能会使用凭证获取对您的存储库的推送和拉取访问权限。如果您所在的系统不安全，则应考虑此风险，并通过省略 `-p` *`password`* 选项并在系统提示时输入密码来以交互方式登录。

3. （可选）如果您有要让映像推送的 Dockerfile，请为新存储库构建并标记映像。将控制台中的 docker build 命令粘贴到终端窗口。确定您与您的 Dockerfile 位于同一目录中。
4. 通过将控制台中的 docker tag 命令粘贴到终端窗口中来为 ECR 注册表和新存储库标记映像。此控制台命令假设您的映像是通过上一步中的 Dockerfile 构建得来的。如果您未通过 Dockerfile 构建映像，请将 *`repository`*`:latest` 的第一个实例更换为要推送的本地映像的映像 ID 或映像名称。
5. 通过将 docker push 命令粘贴到终端窗口中来将新标记的映像推送到 ECR 存储库。
6. 选择 Close。

步骤 2：创建任务定义

任务定义类似于应用程序的蓝图。每次在 Amazon ECS 中启动任务时，您都指定任务定义。这样，服务知道要用于容器的 Docker 映像、任务中要使用的容器数量以及为每个容器分配的资源。

1. 打开 https://console.aws.amazon.com/ecs/home#/firstRun 上的 Amazon ECS 控制台首次运行向导。
2. 从导航栏中，选择美国东部(加利福尼亚北部) 区域。
3. 配置任务定义参数。

   首次运行向导会预加载一个任务定义，您可以看到在控制台中定义的 `simple-app` 容器。您可以选择重命名任务定义，或者查看和编辑容器使用的资源（如 CPU 单元和内存限制）。选择容器名称并编辑显示的值（CPU 单元在 Advanced options (高级选项) 菜单中）。为简单起见，首次运行向导中创建的任务定义仅限用于一个容器。您之后可在 Amazon ECS 控制台中创建多容器任务定义。

   有关每个任务定义参数的作用的更多信息，请参阅任务定义参数 (p. 52)。

   > **Note**
   >
   > 如果在容器定义中使用 Amazon ECR 映像，请确保对 Amazon ECR 映像使用完整的 `registry/repository:tag` 命名。例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*。

4. 选择 Next (下一步)。

步骤 3：配置服务

在向导的此部分中，您可选择您希望配置通过任务定义创建的 Amazon ECS 服务的方式。一个服务将会启动并在您的集群中维护指定数量的任务定义副本。Amazon ECS sample 应用程序是一个基于 Web 的 Hello World–样式应用程序，可无限期地运行。通过将它作为服务运行，它在任务状况不正常或意外停止时会重新启动。

首次运行向导会预加载一个服务定义，您可以看到在控制台中定义的 `sample-app-service` 服务。您可以选择重命名服务，或者通过选择 Edit 并执行以下操作来查看和编辑详细信息：

1. 对于 Service name (服务名称)，为您的服务选择名称。
2. 对于 Desired number of tasks (预期任务数)，输入要使用您指定的任务定义启动的任务数。
3. （可选）您可选择对服务使用 应用程序负载均衡器 负载均衡器。如果从配置为使用负载均衡器的服务启动任务，会向负载均衡器注册任务。来自负载均衡器的流量分布到负载均衡器中的各实例上。有关更多信息，请参阅 Application Load Balancer 简介。

   > **Important**
   >
   > AWS 资源中有 应用程序负载均衡器 时，将产生相应费用。有关更多信息，请参阅 应用程序负载均衡器 定价。

   a. 选择 Application Load Balancer listener port (应用程序负载均衡器侦听器端口)。此处的默认值是为示例应用程序设置的，但您可为负载均衡器配置不同的侦听器选项。有关更多信息，请参阅服务负载均衡 (p. 284)。
   b. 在 Application Load Balancer target group name (Application Load Balancer 目标组名称) 字段中，指定目标组的名称。
   c. 在 Service IAM Role (服务 IAM 角色) 部分，选择您已创建的现有 Amazon ECS 服务 (`ecsServiceRole`) 角色，或选择 Create new role (创建新角色) 为服务创建所需的 IAM 角色。有关更多信息，请参阅 Amazon ECS 服务计划程序 IAM 角色 (p. 410)。

4. 查看服务设置，然后选择 Next step (下一步)。

步骤 4：配置集群

在向导的此部分中，命名您的集群。然后，Amazon ECS 将会为您完成网络和 IAM 配置。

1. 对于 Cluster name (集群名称)，选择集群的名称。

2. 对于 EC2 instance type (EC2 实例类型)，选择要用于您的容器实例的实例类型。具有更多 CPU 和内存资源的实例类型可处理更多任务。有关不同实例类型的更多信息，请参阅 Amazon EC2 实例。
3. 对于 Number of instances (实例数)，键入要在集群中启动以放置任务的 Amazon EC2 实例数。集群中的实例越多，可在其中放置的任务就越多。Amazon EC2 实例在 AWS 资源中存在期间会产生费用。有关更多信息，请参阅 Amazon EC2 定价。

   Note

   如果您创建的服务中有多个将容器端口公开给容器实例端口的所需任务（如 Amazon ECS sample 应用程序），则必须在此处指定相同数量的实例。
4. 选择要用于您的容器实例的密钥对名称。使用 SSH 登录实例时需要此密钥对。如果不在此指定密钥对，则无法使用 SSH 连接到容器实例。如果还没有密钥对，您可以在 Amazon EC2 控制台中创建一个。有关更多信息，请参阅 Amazon EC2 密钥对。
5. （可选）在 Security Group 部分，您可选择一个用于限制对实例的访问的 CIDR 块。默认值 (Anywhere) 允许来自整个 Internet 的访问。
6. 在 Container instance IAM role (容器实例 IAM 角色) 部分，选择您已创建的现有 Amazon ECS 容器实例 (`ecsInstanceRole`) 角色，或选择 Create new role (创建新角色) 为容器实例创建所需的 IAM 角色。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。
7. 选择 Review & Launch (审核并启动)。

步骤 5: 审核

1. 检查任务定义、任务配置和集群配置，然后单击 Create (创建) 完成操作。您会定向到 Launch Status (启动状态) 页面，此页面显示您的启动状态。它描述此过程的每一步（在创建 Auto Scaling 组时，这可能需要几分钟才能完成和填充）。
2. 在启动完成后，选择 View service (查看服务)。

步骤 6：（可选）查看您的服务

如果您的服务是基于 Web 的应用程序 (如 Amazon ECS sample 应用程序)，可以使用 Web 浏览器查看其容器。

1. 在 Service: ***service-name*** (服务: <服务名称>) 页面上，选择 Tasks (任务)。
2. 从您的服务中的任务列表中选择任务。
3. 在 Network (网络) 部分中，选择您的任务的 ENI Id 值。这将转到 Amazon EC2 控制台，您可以在其中查看与任务关联的网络接口的详细信息，包括 IPv4 Public IP (IPv4 公有 IP) 地址。
4. 在 Web 浏览器中，输入 IPv4 Public IP (IPv4 公有 IP) 地址，您应该可以看到显示 Amazon ECS sample 应用程序的网页。

清除 Amazon ECS 资源

完成对特定 Amazon ECS 集群的试验或使用后，您应清除与其关联的资源，以避免产生与您未使用的资源相关的费用。

有些 Amazon ECS 资源（如任务、服务、集群和容器实例）是使用 Amazon ECS 控制台清除的。其他资源（例如 Amazon EC2 实例、Elastic Load Balancing 负载均衡器和 Auto Scaling 组）必须在 Amazon EC2 控制台中手动清除或通过删除创建它们的 AWS CloudFormation 堆栈来清除。

缩减服务

如果您的集群包含任何服务，您应该先将这些服务中的任务的预期数量缩减至 0，以便让 Amazon ECS 不在您清除资源时尝试在您的容器实例上启动新任务。按照更新服务 (p. 326)中的过程操作并在 Number of tasks 字段中输入 0。

或者，您也可以使用以下 AWS CLI 命令缩减您的服务。确保替换区域、集群名称和您要缩减的每个服务的服务名称。

```
aws ecs update-service --cluster default --service service_name --desired-count 0 --
region us-west-2
```

删除服务

您必须先删除集群中的服务才能删除集群。在您的服务缩减至 0 个任务后，您就可以删除服务。对于您的集群中的每个服务，请按照删除服务 (p. 328)中的过程进行删除。

或者，您也可以使用以下 AWS CLI 命令删除服务。确保替换区域、集群名称和您要删除的每个服务的服务名称。

```
aws ecs delete-service --cluster default --service service_name --region us-west-2
```

取消注册容器实例

您必须先取消注册集群中的容器实例才能删除集群。对于您的集群中的每个容器实例，请按照取消注册容器实例 (p. 196)中的过程进行取消注册。

或者，您也可以使用以下 AWS CLI 命令取消注册容器实例。确保替换区域、集群名称和您要取消注册的每个容器实例的容器实例 ID。

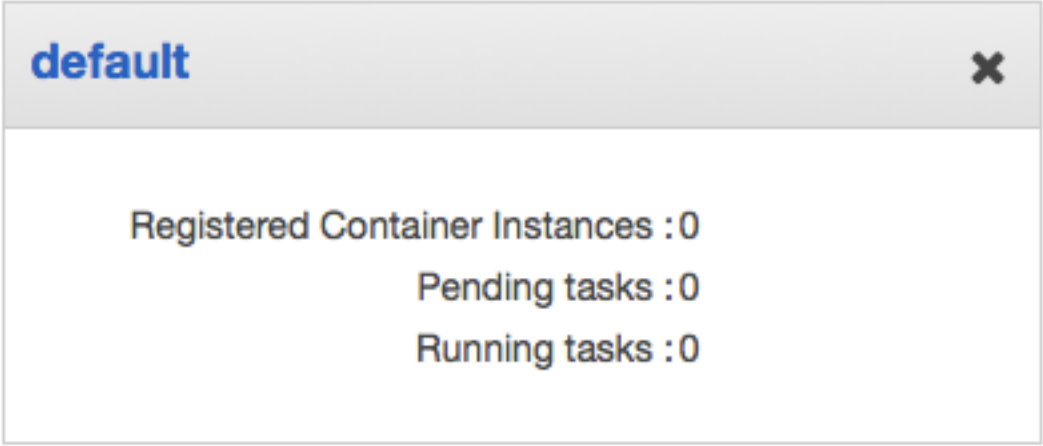
```
aws ecs deregister-container-instance --cluster default --container-
instance container_instance_id --region us-west-2 --force
```

删除集群

从您的 Amazon ECS 集群中删除活动资源后，您就可以删除该集群。

删除集群

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏中，选择集群的正确区域。
3. 在导航窗格中，选择 Clusters。
4. 在 Clusters (集群) 页面上，单击要删除的集群右上角的 x。



5. 选择 Yes, Delete。

或者，您也可以使用以下 AWS CLI 命令删除集群。确保替换区域和您要删除的每个集群的集群名称。

```
aws ecs delete-cluster --cluster default --region us-west-2
```

删除 AWS CloudFormation 堆栈

如果您按照控制台首次运行向导创建了 Amazon ECS 资源，那么您的资源将包含在 AWS CloudFormation 堆栈中。您可以通过删除此堆栈来完全清除与其关联的所有剩余的 AWS 资源。删除 CloudFormation 堆栈将终止 EC2 实例、删除 Auto Scaling 组、删除任何 Elastic Load Balancing 负载均衡器，并删除与集群关联的 Amazon VPC 子网和 Internet 网关。

删除 AWS CloudFormation 堆栈

1. 通过以下网址打开 AWS CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 从导航栏中，选择在其中创建集群的区域。
3. 选择与您的 Amazon ECS 资源关联的堆栈。Stack Name 值以 `EC2ContainerService-default` 开头。
4. 选择删除堆栈，然后选择是，删除。

Amazon ECS 上的 AWS Fargate

AWS Fargate 是可与 Amazon ECS 结合使用的技术，使您在运行容器时不必管理 Amazon EC2 实例的服务器或集群。使用 AWS Fargate，您不必再预置、配置或扩展虚拟机集群即可运行容器。这样一来，您就无需再选择服务器类型、确定扩展集群的时间和优化集群打包。

您在运行使用 Fargate 启动类型的任务和服务时，需要将应用程序打包到容器中、指定 CPU 和内存要求、定义联网和 IAM 策略并启动应用程序。每个 Fargate 任务都具有自己的隔离边界，不与其他任务共享底层内核、CPU 资源、内存资源或弹性网络接口。

本主题介绍 Fargate 任务和服务的不同组件，还列出了将 Fargate 与 Amazon ECS 结合使用时的特别注意事项。

带 Amazon ECS 的 AWS Fargate 目前仅在以下区域可用：

区域名称	区域
美国东部 (弗吉尼亚北部)	us-east-1
美国东部 (俄亥俄)	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部 (俄勒冈)	us-west-2
亚太地区 (孟买)	ap-south-1
欧洲 (爱尔兰)	eu-west-1
欧洲 (伦敦)	eu-west-2
欧洲 (法兰克福)	eu-central-1
亚太地区 (东京)	ap-northeast-1
加拿大 (中部)	ca-central-1
亚太地区 (首尔)	ap-northeast-2
亚太地区 (新加坡)	ap-southeast-1
亚太地区 (悉尼)	ap-southeast-2

下面的演练可帮助您了解如何开始将 AWS Fargate 与 Amazon ECS 配合使用：

- 开始使用 Amazon ECS (p. 19)
- the section called “教程：使用 AWS CLI 创建包含 Fargate 任务的集群” (p. 525)
- the section called “教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群” (p. 433)

任务定义

使用 Fargate 启动类型的任务并不支持所有可用的任务定义参数。某些参数完全不受支持，而其他参数对于 Fargate 任务的行为则不同。

以下任务定义参数在 Fargate 任务中无效：

- `disableNetworking`
- `dnsSearchDomains`
- `dnsServers`
- `dockerSecurityOptions`
- `extraHosts`
- `gpu`
- `ipcMode`
- `links`
- `pidMode`
- `placementConstraints`
- `privileged`
- `systemControls`

以下任务定义参数在 Fargate 任务中有效，但应注意以下限制：

- `linuxParameters` – 当指定适用于容器的 Linux 特定选项时，对于 `capabilities`，不支持 `add` 参数。`devices`、`sharedMemorySize` 和 `tmpfs` 参数不受支持。有关更多信息，请参阅Linux 参数 (p. 69)。
- `volumes` – Fargate 任务仅支持绑定装载主机卷，因此， `dockerVolumeConfiguration` 参数不受支持。有关更多信息，请参阅卷 (p. 74)。

为确保您的任务定义可验证能否用于 Fargate 启动类型，可在注册任务定义时指定以下内容：

- 在 AWS 管理控制台中，为 Requires Compatibilities (需要兼容性) 字段指定 `FARGATE`。
- 在 AWS CLI 中指定 `--requires-compatibilities` 选项。
- 在 Amazon ECS API 中指定 `requiresCompatibilities` 标记。

网络模式

Fargate 任务定义需要将网络模式设置为 `awsvpc`。`awsvpc` 网络模式会为每个任务提供它自己的弹性网络接口。有关更多信息，请参阅 `awsvpc` 网络模式的任务联网 (p. 95)。

创建服务或手动运行任务时还需要网络配置。有关更多信息，请参阅 任务联网 (p. 32)。

任务 CPU 和内存

Fargate 任务定义需要您在任务级别指定 CPU 和内存。尽管您还可以在容器级别为 Fargate 任务指定 CPU 和内存，但这是可选项。只需在任务级别指定这些资源便可满足大多数使用案例。下表显示了任务级 CPU 和内存的有效组合。

CPU 值	内存值
256 (.25 vCPU)	0.5GB、1GB、2GB
512 (.5 vCPU)	1GB、2GB、3GB、4GB
1024 (1 vCPU)	2GB、3GB、4GB、5GB、6GB、7GB、8GB
2048 (2 vCPU)	4GB 到 16GB 之间（以 1GB 为增量）

CPU 值	内存值
4096 (4 vCPU)	8GB 到 30GB 之间（以 1GB 为增量）

日志记录

Fargate 任务定义只支持用于日志配置的 `awslogs` 和 `splunk` 日志驱动程序。

`awslogs` 日志驱动程序会将您的 Fargate 任务配置为向 Amazon CloudWatch Logs 发送日志信息。下面显示了任务定义中配置 awslogs 日志驱动程序的代码段：

```
"logConfiguration": {
   "logDriver": "awslogs",
   "options": {
      "awslogs-group" : "/ecs/fargate-task-definition",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
}
```

有关在任务定义中使用 `awslogs` 日志驱动程序以将容器日志发送到 CloudWatch Logs 的更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。

有关在任务定义中使用 `splunk` 日志驱动程序的更多信息，请参阅示例：splunk 日志驱动程序 (p. 114)。

Amazon ECS 任务执行 IAM 角色

可以使用 Fargate 指定一个可选的任务执行 IAM 角色，以允许您的 Fargate 任务对 Amazon ECR 进行 API 调用。这些 API 调用会拉取容器映像并调用 CloudWatch 来存储容器应用程序日志。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

示例任务定义

下面是用于设置 Web 服务器的示例任务定义（使用 Fargate 启动类型）：

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!
</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></
html>' >  /usr/local/apache2/htdocs/index.html && httpd-foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
```

```
            "name": "sample-fargate-app",
            "portMappings": [
               {
                  "containerPort": 80,
                  "hostPort": 80,
                  "protocol": "tcp"
               }
            ]
         }
      ],
      "cpu": "256",
      "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
      "family": "fargate-task-definition",
      "memory": "512",
      "networkMode": "awsvpc",
      "requiresCompatibilities": [
          "FARGATE"
       ]
}
```

任务存储

在预配置时，每个 Fargate 任务都会收到以下存储。任务存储是短暂存储。Fargate 任务停止后，该存储将被删除。

- 10 GB 的 Docker 层存储
- 额外 4 GB 用于卷挂载。可在任务定义中使用 `volumes`、`mountPoints` 和 `volumesFrom` 参数在容器之间挂载和共享此存储。

  Note

  `host` 和 `sourcePath` 参数不受支持。

有关 Amazon ECS 默认服务限制的更多信息，请参阅Amazon ECS 服务限制 (p. 540)。

下面显示任务定义中两个容器共享单个卷的代码段：

```
{
   "containerDefinitions": [
      {
         "image": "my-repo/database",
         "mountPoints": [
            {
               "containerPath": "/var/scratch",
               "sourceVolume": "database_scratch"
            }
         ],
         "name": "database1",
      },
      {
         "image": "my-repo/database",
         "mountPoints": [
            {
               "containerPath": "/var/scratch",
               "sourceVolume": "database_scratch"
            }
         ],
         "name": "database2",
      }
   ],
   "volumes": [
```

```
        {
            "name": "database_scratch"
        }
    ]
}
```

任务和服务

在准备好 Fargate 任务定义后，需要在创建服务时制定一些决策。

任务联网

使用 Fargate 启动类型的任务需要 `awsvpc` 网络模式，该模式可为每个任务提供弹性网络接口。在使用此网络模式运行任务或创建服务时，必须指定一个或多个要附加网络接口的子网以及一个或多个要应用于该网络接口的安全组。

如果您使用的是公有子网，请决定是否为网络接口提供公有 IP 地址。对于公有子网中拉取容器映像的 Fargate 任务，需要向该任务的弹性网络接口分配一个公有 IP 地址，还需要一个到 Internet 的路由，或一个可以将请求路由到 Internet 的 NAT 网关。对于私有子网中拉取容器映像的 Fargate 任务，私有子网需要连接 NAT 网关以将请求路由到 Internet。有关更多信息，请参阅 `awsvpc` 网络模式的任务联网 (p. 95)。

下面是 Fargate 服务的 networkConfiguration 部分示例：

```
"networkConfiguration": {
   "awsvpcConfiguration": {
      "assignPublicIp": "ENABLED",
      "securityGroups": [ "sg-12345678" ],
      "subnets": [ "subnet-12345678" ]
   }
}
```

具有使用 `awsvpc` 网络模式的任务（例如，那些 Fargate 启动类型的任务）的服务仅支持 应用程序负载均衡器 和 网络负载均衡器；不支持 传统负载均衡器。此外，当您为这些服务创建任何目标组时，必须选择 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。有关更多信息，请参阅服务负载均衡 (p. 284)。

私有注册表身份验证

Fargate任务可使用基本身份验证对私有注册表（包括 Docker Hub）进行身份验证。当启用私有注册表身份验证时，您可以在任务定义中使用私有 Docker 映像。

要使用私有注册表身份验证，需要使用 AWS Secrets Manager 为私有注册表创建一个包含您的凭证的密钥。然后，在容器定义中，通过您创建的密钥的完整 ARN 指定 `repositoryCredentials`。下面是显示必需参数的任务定义代码段：

```
"containerDefinitions": [
   {
      "image": "private-repo/private-image",
      "repositoryCredentials": {
         "credentialsParameter:
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      }
   }
]
```

有关更多信息，请参阅 任务的私有注册表身份验证 (p. 104)。

集群

集群可以包含使用 Fargate 和 EC2 启动类型的任务。在 AWS 管理控制台中查看您的集群时，会分别显示 Fargate 和 EC2 任务计数。

有关 Amazon ECS 集群的更多信息，包括创建集群的演练，请参阅Amazon ECS 集群 (p. 38)。

Fargate 任务停用

在 AWS 检测到托管任务的底层硬件发生无法弥补的故障时或需要针对安全性问题打补丁时，将停用 Fargate 任务。大多数安全补丁程序以透明方式处理，无需执行任何操作或者重新启动您的任务。但对于某些问题，可能需要重新启动该任务。

当任务到达其计划的停用日期时，AWS 会将其停止或终止。如果任务是服务的一部分，则任务会自动停止，服务计划程序会启动新任务来代替它。如果您使用的是独立任务，您将收到任务停用通知。有关更多信息，请参阅 任务停用 (p. 267)。

AWS Fargate 平台版本

AWS Fargate 平台版本用于指代 Fargate 任务基础设施的特定运行时环境。它是内核和容器运行时版本的组合。

随着运行时环境的发展，例如，如果有内核或操作系统更新、新功能、错误修复或安全更新，将会发布新平台版本。Fargate 任务的安全更新和修补程序将自动部署。如果安全问题影响平台版本，AWS 为平台版本提供补丁。在某些情况下，您可能会收到已计划停止您的 Fargate 任务的通知。有关更多信息，请参阅 任务停用 (p. 267)。

主题

- 平台版本注意事项 (p. 33)
- 可用的 AWS Fargate 平台版本 (p. 33)

平台版本注意事项

指定平台版本时应该考虑以下事项：

- 指定平台版本时，您可以使用版本号 (例如 `1.2.0`) 或 `LATEST`。
- 要使用特定平台版本，请在创建或更新服务时指定版本号。如果指定 `LATEST`，则您的任务将使用目前可用的最近平台版本，这可能不是最新的平台版本。
- 如果您的服务有正在运行的任务并希望更新其平台版本，您可以更新服务，指定新平台版本，然后选择强制实施新部署。您的任务使用最新平台版本重新部署。有关更多信息，请参阅更新服务 (p. 326)。
- 如果您的服务扩展而没有更新平台版本，这些任务将收到在服务的当前部署中指定的平台版本。

可用的 AWS Fargate 平台版本

以下是当前可用的平台版本的列表：

Fargate 平台版本-1.3.0

- 增加了 Fargate 任务的任务回收，此过程用于刷新作为 Amazon ECS 服务一部分的任务。有关更多信息，请参阅Fargate 任务回收 (p. 268)。
- 从 2019 年 3 月 27 日开始，所启动的任何新的 Fargate 任务均可以使用其他任务定义参数，您可以通过这些参数定义代理配置、容器启动和关闭的依赖条件，以及每个容器的启动和停止超时值。有关更多信息，请参阅代理配置 (p. 79)、容器依赖项 (p. 71)和容器超时 (p. 72)。
- 从 2019 年 4 月 2 日开始，所启动的任何新的 Fargate 任务均支持向容器中注入敏感数据，方式是将您的敏感数据存储在 AWS Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数中，然后在容器定义中引用它们。有关更多信息，请参阅指定敏感数据 (p. 107)。
- 从 2019 年 5 月 1 日开始，所启动的任何新的 Fargate 任务均支持使用 `secretOptions` 容器定义参数来引用容器的日志配置中的敏感数据。有关更多信息，请参阅指定敏感数据 (p. 107)。
- 从 2019 年 5 月 1 日开始，所启动的任何新的 Fargate 任务均支持 `splunk` 日志驱动程序以及 `awslogs` 日志驱动程序。有关更多信息，请参阅存储和日志记录 (p. 63)。
- 从 2016 年 7 月 9 日开始，启动的任何新 Fargate 任务都支持 CloudWatch Container Insights。有关更多信息，请参阅Amazon ECS CloudWatch Container Insights (p. 363)。

Fargate 平台版本-1.2.0

- 添加了对使用 AWS Secrets Manager 进行私有注册表身份验证的支持。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。

Fargate 平台版本-1.1.0

- 添加了对 Amazon ECS 任务元数据终端节点的支持。有关更多信息，请参阅 Amazon ECS 任务元数据终端节点 (p. 236)。
- 添加了对容器定义中 Docker 运行状况检查的支持。有关更多信息，请参阅 运行状况检查 (p. 57)。
- 添加了对 Amazon ECS 服务发现的支持。有关更多信息，请参阅服务发现 (p. 306)。

Fargate 平台版本-1.0.0

- 基于 Amazon Linux 2017.09。
- 首次发布。

在 Amazon ECS 上使用 GPU

Amazon ECS 通过使您能够使用支持 GPU 的容器实例创建集群来支持利用 GPU 的工作负载。使用 p2 和 p3 实例类型的基于 GPU 的 Amazon EC2 容器实例提供对 NVIDIA GPU 的访问权限。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Linux 加速计算实例。

Amazon ECS 提供了经过 GPU 优化的 AMI，后者附带了预配置的 NVIDIA 内核驱动程序和 Docker GPU 运行时。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

您可以在任务定义中指定多个 GPU，以便在容器级别考虑任务放置。Amazon ECS 将安排可用的支持 GPU 的容器实例并将物理 GPU 固定到正确的容器以获得最佳性能。

支持以下基于 GPU 的 Amazon EC2 实例类型。有关更多信息，请参阅 Amazon EC2 P2 实例、Amazon EC2 P3 实例和 Amazon EC2 G3 实例。

实例类型	GPU	GPU 内存 (GiB)	vCPU	内存 (GiB)
p2.xlarge	1	12	4	61
p2.8xlarge	8	96	32	488
p2.16xlarge	16	192	64	732
p3.2xlarge	1	16	8	61
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768
g3s.xlarge	1	8	4	30.5
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488

主题

- 有关使用 GPU 的注意事项 (p. 35)
- 在任务定义中指定 GPU (p. 36)

有关使用 GPU 的注意事项

在 Amazon ECS 上使用 GPU 之前，请注意以下事项：

- 您的集群可以包含 GPU 和非 GPU 容器实例的组合。
- 在运行任务或创建服务时，您可以在配置任务放置约束时使用实例类型属性，以确保在适当的容器实例上启动任务。这使您能够有效地使用您的资源。有关更多信息，请参阅Amazon ECS 任务放置 (p. 255)。

  以下示例在您的默认集群中的 `p2.xlarge` 容器实例上启动一个任务。

```
aws ecs run-task --cluster default --task-definition ecs-gpu-task-def \
--placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 p2.xlarge" --region us-east-2
```

- 在使用 Docker CLI 时，`--runtime` 参数可用于在运行容器时指定 NVIDIA 运行时。例如：

```
docker run --runtime=nvidia --rm nvidia/cuda:9.0-base nvidia-smi
```

在任务定义中指定 GPU

要在容器实例和 Docker GPU 运行时上利用 GPU，请确保在任务定义中指定容器所需的 GPU 数量。由于已放置支持 GPU 的容器，因此 Amazon ECS 容器代理会将所需数量的物理 GPU 固定到相应的容器中。为某个任务中的所有容器预留的 GPU 的数量不应超过该任务在其上启动的容器实例的可用 GPU 的数量。有关更多信息，请参阅创建任务定义 (p. 45)。

> **Important**
>
> 如果任务定义中未指定 GPU 要求，则任务将使用默认 Docker 运行时。

下面显示了任务定义中的 GPU 要求的 JSON 格式：

```
{
  "containerDefinitions": [
     {
        ...
        "resourceRequirements" : [
            {
               "type" : "GPU",
               "value" : "2"
            }
        ],
     },
...
}
```

以下示例演示了指定 GPU 要求的 Docker 容器的语法。此容器使用 2 个 GPU，运行 `nvidia-smi` 实用程序，然后退出。

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:9.0-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
```

```
  ],
  "family": "example-ecs-gpu"
}
```

Amazon ECS 集群

Amazon ECS 集群是任务或服务的逻辑分组。 如果您在运行使用 EC2 启动类型的任务或服务，则集群也是容器实例的分组。 在您首次使用 Amazon ECS 时，系统将为您创建一个默认集群，但您可以在一个账户中创建多个集群以保持您的资源独立。

以下是有关 Amazon ECS 集群的一般概念。

- 集群是特定于区域的。
- 集群可以包含使用 Fargate 和 EC2 启动类型的任务。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。
- 对于使用 EC2 启动类型的任务，集群可以包含多个不同的容器实例类型，但每个容器实例一次只能属于一个集群。
- 您可以为集群创建自定义 IAM 策略以允许或限制用户对特定集群的访问。有关更多信息，请参阅 集群示例 (p. 379)中的Amazon Elastic Container Service 基于身份的策略示例 (p. 374)部分。

主题

- 创建集群 (p. 38)
- 扩展集群 (p. 40)
- 删除集群 (p. 41)

创建集群

您可以使用 AWS 管理控制台创建 Amazon ECS 集群，如本主题中所述。开始之前，请确保您已完成Amazon ECS 的设置 (p. 7)中的步骤。如果您正在启动使用 EC2 启动类型的任务，在创建集群后，您可以在集群中注册容器实例。

> **Note**
>
> 此群集创建向导提供了一种简单的方法来创建 Amazon ECS 集群所需的资源。它还可让您自定义几个常用的集群配置选项。但是，该向导不允许您自定义每个资源选项。例如，您无法使用该向导自定义容器实例 AMI ID。 如果您的需求超出此向导中支持的范围，请考虑使用我们在 https://github.com/awslabs/ecs-refarch-cloudformation 上的参考架构。
> 不要试图在该向导创建基础资源后直接修改这些资源。

创建集群

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Clusters。
4. 在 Clusters 页面上，选择 Create Cluster。
5. 对于 Select cluster compatibility (选择集群兼容性)，选择以下选项之一，然后选择 Next Step (下一步)：
   - Networking only (仅联网)– 通过此选项，您可以使用 Fargate 启动任务集群。Fargate 启动类型允许您运行容器化的应用程序，而无需预置和管理后台基础设施。在注册您的任务定义时，Fargate 会为您启动容器。
   - EC2 Linux+Networking (EC2 Linux+联网)– 通过此选项，您可以使用 EC2 启动类型和 Linux 容器启动任务集群。EC2 启动类型允许您在管理的 Amazon EC2 实例集群上运行容器化的应用程序。
   - EC2 Windows + Networking (EC2 Windows + 联网) – 通过此选项，您可以使用 EC2 启动类型通过 Windows 容器启动任务集群。EC2 启动类型允许您在管理的 Amazon EC2 实例集群上运行容器化的应用程序。有关更多信息，请参阅 Windows 容器 (p. 599)。

如果您选择 Networking only (仅联网) 集群模板，请继续执行以下步骤。否则，您可以跳到下一部分。

使用 Networking only (仅联网) 集群模板

1. 在 Configure cluster (配置集群) 页面上，输入 Cluster name (集群名称)。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
2. 在 Networking (联网) 部分中，为您的集群配置 VPC。您可以保留默认设置，也可以通过以下步骤修改这些设置。
   a. （可选）如果您选择创建新 VPC，请为 CIDR Block (CIDR 块) 选择 VPC 的 CIDR 块。有关更多信息，请参阅 Amazon VPC 用户指南 中的您的 VPC 和子网。
   b. 对于 Subnets (子网)，选择要用于 VPC 的子网。您可以保留默认设置，也可以进行修改以满足您的需求。
3. 在 Tags (标签) 部分中，为要与集群关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
4. 在 CloudWatch Container Insights 部分中，选择是否为集群启用 Container Insights。有关更多信息，请参阅Amazon ECS CloudWatch Container Insights (p. 363)。
5. 选择 Create。

如果您选择 EC2 Linux + Networking (EC2 Linux + 联网) 或 EC2 Windows + Networking (EC2 Windows + 联网) 模板，请完成以下步骤：

使用 EC2 Linux + Networking (EC2 Linux + 联网) 或 EC2 Windows + Networking (EC2 Windows + 联网) 集群模板

1. 对于 Cluster name (集群名称)，输入集群的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
2. （可选）要创建没有任何资源的集群，请依次选择 Create an empty cluster (创建空集群) 和 Create (创建)。
3. 对于 Provisioning model (预配置模式)，请选择下列实例类型之一：
   - On-Demand Instance (按需实例) – - 使用按需实例，您只需要按小时支付计算容量费用，无需长期购买或预付款。
   - Spot– 使用 Spot 实例，您可以为空闲的 Amazon EC2 计算容量出价，最多可低至按需价格的 1 折。有关更多信息，请参阅 Spot 实例。

     Note

     Spot 实例可能会中断。我们建议您不要为不能中断的应用程序使用 Spot 实例。有关更多信息，请参阅 Spot 实例中断。
4. 对于 Spot 实例，执行以下操作；否则跳到下一步。
   a. 对于 Spot Instance allocation strategy，请选择满足您需求的策略。有关更多信息，请参阅 Spot 队列分配策略。
   b. 对于 Maximum bid price (per instance/hour)，指定一个出价。如果您的出价低于所选实例类型的 Spot 价格，则您的 Spot 实例不会启动。
5. 对于 EC2 instance type (EC2 实例类型)，选择容器实例的 Amazon EC2 实例类型。您选择的实例类型决定了可用于您的任务的 EC2 AMI ID 和资源。对于 GPU 工作负载，从 P2 或 P3 实例系列选择实例类型。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。
6. 对于 Number of instances (实例数量)，选择要在集群中启动的 EC2 实例的数量。这些实例是使用您选择的实例类型所需的最新的 Amazon ECS-optimized Amazon Linux AMI 启动的。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。
7. 对于 EC2 AMI Id，为您的容器实例选择 Amazon ECS 优化的 AMI。可用的 AMI 将取决于您选择的区域和 EC2 实例类型。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

8. 对于 EBS storage (GiB) (EBS 存储(GiB))，选择容器实例上要用作数据存储的 Amazon EBS 卷的大小。您可增加数据卷的大小，以支持更大的映像和容器存储。
9. 对于 Key pair (密钥对)，选择要用于容器实例进行 SSH 访问的 Amazon EC2 密钥对。如果您未指定密钥对，则无法使用 SSH 连接到容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 密钥对。
10. 在 Networking 部分中，配置要启动容器实例的 VPC。默认情况下，集群创建向导会创建一个新的 VPC (具有位于不同可用区中的两个子网) 和一个安全组 (在端口 80 上对 Internet 开放)。这是一个适用于 HTTP 访问的基本设置。不过，您可以通过执行以下子步骤来修改这些设置。

    a. 对于 VPC，创建一个新 VPC 或选择现有 VPC。
    b. (可选) 如果您选择创建新 VPC，请为 CIDR Block 选择 VPC 的 CIDR 块。有关更多信息，请参阅 Amazon VPC 用户指南 中的您的 VPC 和子网。
    c. 对于 Subnets (子网)，选择要用于 VPC 的子网。如果您选择创建新 VPC，则可保留默认设置，也可修改这些设置以满足您的需求。如果选择使用现有 VPC，则在该 VPC 中选择一个或多个要用于集群的子网。
    d. 对于 Security group，选择要附加到集群中的容器实例的安全组。如果您选择创建新的安全组，可指定一个 CIDR 块以允许来自它的入站流量。默认端口 0.0.0.0/0 向 Internet 开放。您也可以选择要在容器实例上开放的一个端口或一系列连续端口。要实现更复杂的安全组规则，您可以选择已创建的现有安全组。

        Note

        您还可以选择在创建集群后创建新的安全组并修改规则。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的适用于 Linux 实例的 Amazon EC2 安全组。

    e. 在 Container instance IAM role (容器实例 IAM 角色) 部分中，选择要用于容器实例的 IAM 角色。如果您的账户具有在首次运行向导中为您创建的 ecsInstanceRole，则默认情况下会将其选定。如果您的账户中没有此角色，则可选择创建此角色，也可以选择另一个用于您的容器实例的 IAM 角色。

        Important

        如果未使用适当的 IAM 权限启动容器实例，则 Amazon ECS 代理不会连接到集群。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

    f. 如果之前选择了 Spot 实例类型，则 Spot Fleet Role IAM role (Spot 队列角色 IAM 角色) 部分指示会创建一个 IAM 角色 `ecsSpotFleetRole`。
    g. 在 Tags (标签) 部分中，为要与集群关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
    h. 在 CloudWatch Container Insights 部分中，选择是否为集群启用 Container Insights。有关更多信息，请参阅Amazon ECS CloudWatch Container Insights (p. 363)。
    i. 选择 Create。

扩展集群

如果集群包含 Amazon EC2 容器实例，以下操作将帮助您扩展集群中的 Amazon EC2 实例数。

Note

可以使用 服务 Auto Scaling 扩展包含 Fargate 任务的集群。有关更多信息，请参阅服务 Auto Scaling (p. 299)。

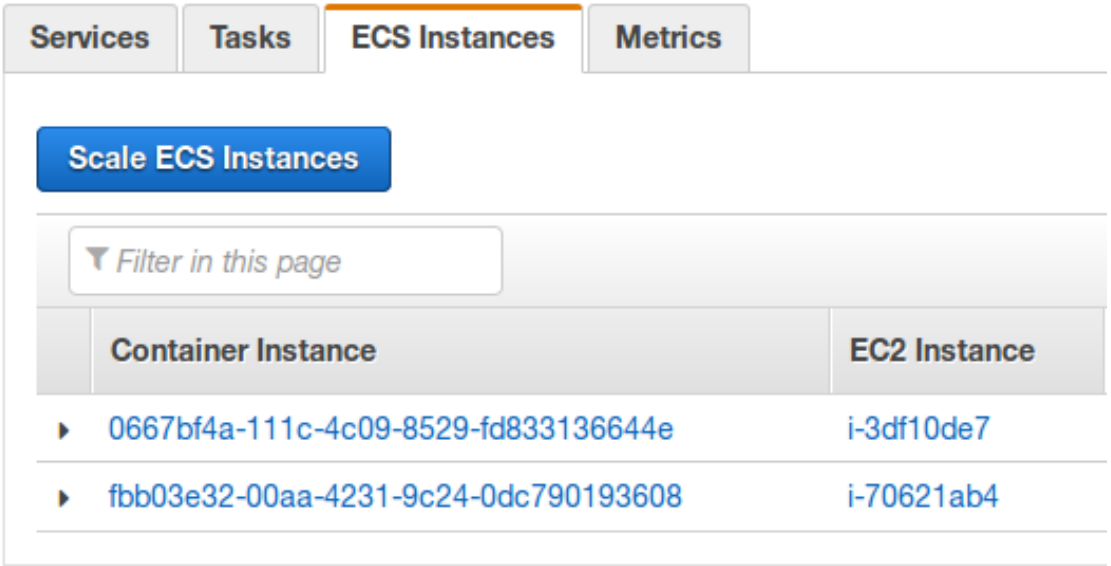
如果集群是在 2015 年 11 月 24 日后通过控制台首次运行体验创建的，则可以扩展或缩减与为集群创建的与 AWS CloudFormation 堆栈关联的 Auto Scaling 组来添加或删除容器实例。您可以从 Amazon ECS 控制台内部执行此扩展操作。

如果您的集群不是在 2015 年 11 月 24 日后通过控制台首次运行体验创建的，则不能从 Amazon ECS 控制台扩展集群。但是，您仍可以在 Auto Scaling 控制台中修改与集群关联的现有 Auto Scaling 组。如果您没有

与集群关联的 Auto Scaling 组，则可从现有容器实例中创建一个组。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南 中的使用 EC2 实例创建 Auto Scaling 组。您也可以从 Amazon EC2 控制台手动启动或终止容器实例；有关更多信息，请参阅启动 Amazon ECS 容器实例 (p. 165)。

扩展集群

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏中，选择集群所在的区域。
3. 在导航窗格中，选择 Clusters (集群) 并选择要扩展的集群。
4. 在 Cluster : *name* 页面上，选择 ECS Instances。



如果出现 Scale ECS Instances 按钮，则可以在下一步骤中扩展您的集群。否则，必须手动调整 Auto Scaling 组以向上扩展或向下扩展实例，也可以在 Amazon EC2 控制台中手动启动或终止容器实例。

5. 选择 Scale ECS Instances。
6. 对于 Desired number of instances (所需的实例数)，输入要将群集扩展到的实例的数量，然后选择 Scale (扩展)。

   Note

   如果您要减少集群中容器实例的数量，则在达到所需的数量之前，随机选择的容器实例会被终止，并且在已终止的实例上运行的任何任务也会被停止。

删除集群

使用完集群后，可以将其删除。在 Amazon ECS 控制台中删除集群时，随集群一起删除的相关资源因创建集群的方式而异。下面过程中的Step 5 (p. 42) 随相应条件会有所变化。

如果您的集群是 2015 年 11 月 24 日后使用开始使用 Amazon ECS (p. 19) 中描述的控制台首次运行体验创建的，或者是使用创建集群 (p. 38)中描述的集群创建向导创建的，则在删除集群时，还将删除为集群创建的 AWS CloudFormation 堆栈。

如果您的集群是在 2015 年 11 月 24 日之前手动创建的（在没有集群创建向导的情况下）或者是使用控制台首次运行体验创建的，则必须先取消注册（或终止）与集群关联的任何容器实例，然后才能删除集群。有关更多信息，请参阅 取消注册容器实例 (p. 196)。这种情况下，删除集群后，应删除与集群关联的所有剩余

AWS CloudFormation 堆栈资源或 Auto Scaling 组，以免这些资源以后产生任何费用。有关更多信息，请参阅 删除 AWS CloudFormation 堆栈 (p. 27)。

删除集群

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Clusters。
4. 在 Clusters 页面上，选择要删除的集群。

   Note

   如果您的集群已注册容器实例，则必须取消注册或终止这些实例。有关更多信息，请参阅 取消注册容器实例 (p. 196)。
5. 选择 Delete Cluster。您会看到两条确认提示之一：

   - Deleting the cluster also deletes the CloudFormation stack EC2ContainerService-***cluster_name (######## CloudFormation ## EC2ContainerService-cluster_name)*** – 删除此群集将清除已使用此群集创建的相关资源，包括 Auto Scaling 组、VPC 或负载均衡器。
   - Deleting the cluster does not affect CloudFormation resources (删除集群不影响 CloudFormation 资源) – : 删除此群集不会清除与此群集关联的任何资源，包括 Auto Scaling 组、VPC 或负载均衡器。同样，必须先取消注册或终止注册到此集群的所有容器实例，然后才能删除集群。有关更多信息，请参阅 取消注册容器实例 (p. 196)。您可以访问 https://console.aws.amazon.com/cloudformation/ 上的 AWS CloudFormation 控制台以更新或删除这些资源中的任意资源。有关更多信息，请参阅 删除 AWS CloudFormation 堆栈 (p. 27)。

Amazon ECS 任务定义

需要任务定义才能在 Amazon ECS 中运行 Docker 容器。可在任务定义中指定的一些参数包括：

- 要用于任务中的每个容器的 Docker 映像
- 要用于每个任务或任务中的每个容器的 CPU 和内存数量
- 要使用的启动类型，这决定了托管任务的基础设施。
- 在您的任务中用于容器的 Docker 联网模式
- 要用于您的任务的日志记录配置
- 在容器完成或失败时，任务是否应继续运行
- 容器在启动时应运行的命令
- 应在任务中用于容器的任何数据卷
- 您的任务应使用的 IAM 角色

您可以在任务定义中定义多个容器。您使用的参数取决于您为任务选择的启动类型。并非所有参数都有效。有关可用参数以及在任务定义中有效的启动类型的更多信息，请参阅 任务定义参数 (p. 52)。

您的整个应用程序堆栈不需要存在于单个任务定义上，并且在大多数情况下，它不应存在于单个任务定义上。您的应用程序可通过将相关容器组合到其自己的任务定义（每个任务定义表示一个组件）中来跨多个任务定义。有关更多信息，请参阅 应用程序架构 (p. 43)。

主题

- 应用程序架构 (p. 43)
- 创建任务定义 (p. 45)
- 任务定义参数 (p. 52)
- Amazon ECS 启动类型 (p. 81)
- 在任务中使用数据卷 (p. 83)
- awsvpc 网络模式的任务联网 (p. 95)
- 使用 awslogs 日志驱动程序 (p. 97)
- 任务的私有注册表身份验证 (p. 104)
- 指定敏感数据 (p. 107)
- 示例任务定义 (p. 112)
- 更新任务定义 (p. 118)
- 取消注册任务定义 (p. 118)

应用程序架构

如何在 Amazon ECS 上构建您的应用程序取决于几个因素，其中您使用的启动类型是一个关键因素。我们给出以下指导，按启动类型分类，应该能在操作过程中提供帮助。

使用 Fargate 启动类型

使用 Fargate 启动类型为您的任务构建应用程序时，主要问题是什么时候应该把多个容器放到同一个任务定义中，而不是分别在多个任务定义中部署容器。

在符合以下条件时，您应在同一任务定义中放置多个容器：

- 容器共享公共生命周期 (即，它们应该一起启动和终止).

- 容器需要在相同的底层主机上运行（即一个容器在 localhost 端口上引用另一个容器）。
- 您希望您的容器共享资源.
- 您的容器共享数据卷。

否则，您应该在单独的任务定义中定义您的容器，以便您可以分别对其进行扩展、预置和取消预置。

使用 EC2 启动类型

当您考虑如何使用 EC2 启动类型对任务定义和服务建模时，它可以帮助考虑哪些流程需要一起运行以及您将如何扩展每个组件。

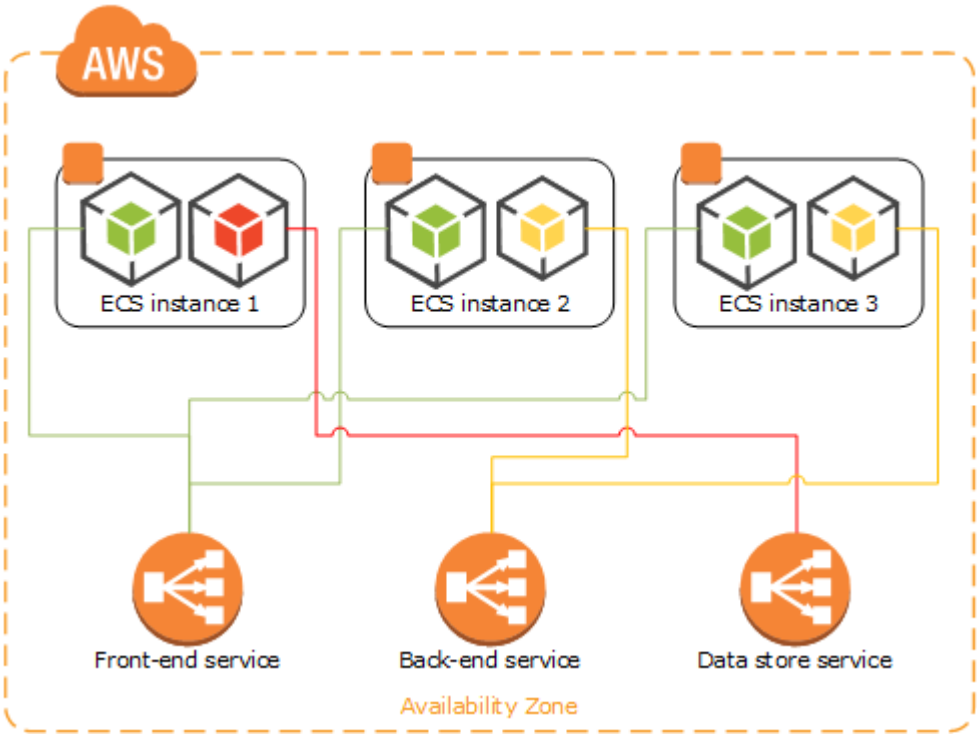
例如，假设某个应用程序包含以下组件：

- 一项在网页上显示信息的前端服务
- 一项为前端服务提供 API 的后端服务
- 数据存储

在您的开发环境中，您可能要在 Docker 主机上同时运行所有三个容器。您可能倾向于对生产环境使用同一方法，但此方法具有以下几个缺点：

- 对一个组件的更改可能影响所有三个组件，更改的范围可能大于您所预期的范围.
- 每个组件更难以扩展，因为您必须按比例扩展每个容器.
- 任务定义只能具有 10 个容器定义，并且您的应用程序堆栈现在或将来可能需要更多容器.
- 任务定义中的每个容器必须登录同一容器实例，这可能会将您的实例选择限制为最大大小.

相反，您应创建将对用于相同目的的容器进行分组的任务定义，并将不同的组件分成多个任务定义。在此示例中，三个任务定义分别指定一个容器。以下示例集群具有已注册到三个前端服务容器、两个后端服务容器和一个数据存储服务容器的三个容器实例。



您可以对任务定义中的相关容器（例如，必须一起运行的已链接的容器）进行分组。例如，您可以将日志流容器添加到前端服务，并将其包含在同一任务定义中。

在拥有任务定义后，您可以从这些任务定义创建服务以保持所需任务的可用性。有关更多信息，请参阅 创建服务 (p. 317)。在您的服务中，可以将容器与 Elastic Load Balancing 负载均衡器关联。有关更多信息，请参阅 服务负载均衡 (p. 284)。当您的应用程序要求发生更改时，您可以更新服务以增大或减小所需任务数，或者在任务中部署更新版本的容器。有关更多信息，请参阅 更新服务 (p. 326)。

创建任务定义

您必须先创建任务定义，然后才能在 Amazon ECS 上运行 Docker 容器。您可以在任务定义中定义多个容器和数据卷。有关在任务定义中可用参数的更多信息，请参阅 任务定义参数 (p. 52)。

创建新的任务定义

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，依次选择 Task Definitions (任务定义) 和 Create new Task Definition (创建新任务定义)。
3. 在 Select compatibilities 页面上，选择您的任务应使用的启动类型，然后选择 Next step。

   Note

   Fargate 启动类型与 Windows 容器不兼容。
4. 根据您选择的启动类型，按照下列一个选项卡下的步骤操作。

Fargate launch type

使用 Fargate 启动类型兼容性模板

如果选择 Fargate，请完成以下步骤：

1. (可选) 如果您有任务定义的 JSON 表示形式，请完成以下步骤：
   a. 在 Configure task and container definitions 页面上，滚动到页面底部并选择 Configure via JSON。
   b. 将您的任务定义 JSON 粘贴到文本区域并选择 Save。
   c. 验证您的信息并选择 Create。

   滚动到页面底部，选择 Configure via JSON (通过 JSON 配置)。
2. 在 Task Definition Name (任务定义名称) 中，为任务定义键入名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
3. （可选）对于 Task Role (任务角色)，选择 IAM 角色，该角色可提供权限，允许任务中的容器代表您调用 AWS API 操作。有关更多信息，请参阅 任务的 IAM 角色 (p. 418)。

   Note

   此处仅显示具有 Amazon EC2 Container Service Task Role 信任关系的角色。有关如何为任务创建 角色的更多信息，请参阅 。
4. 对于 Task execution IAM role (任务执行 IAM 角色)，选择您的任务执行角色，或者选择 Create new role (新建角色)，以便控制台可以为您创建一个。有关更多信息，请参阅 。
5. 对于 Task size，选择 Task memory (GB) 和 Task CPU (vCPU) 的值。下表显示了有效组合。

CPU 值	内存值
256 (.25 vCPU)	512 MB、1 GB、2 GB
512 (.5 vCPU)	1GB、2GB、3GB、4GB
1024 (1 vCPU)	2GB、3GB、4GB、5GB、6GB、7GB、8GB
2048 (2 vCPU)	4GB 到 16GB 之间 (以 1GB 为增量)
4096 (4 vCPU)	8 GB 到 30 GB（以 1 GB 为增量）

6. 对于您的任务定义中的每个容器，请完成以下步骤：
   a. 选择 Add container (添加容器)。
   b. 填写每个必填字段以及要在您的容器定义中使用的任何可选字段。Advanced container configuration (高级容器配置) 菜单中提供了更多容器定义参数。有关更多信息，请参阅 任务定义参数 (p. 52)。
   c. 选择 Add (添加)，将容器添加到任务定义。
7. （可选）对于 Service Integration (服务集成)，要配置 App Mesh 集成的参数，请选择 Enable App Mesh integration (启用 App Mesh 集成)，然后执行以下操作：
   a. 对于 Application container name (应用程序容器名称)，选择要用于 App Mesh 应用程序的容器名称。必须已在任务定义中定义此容器。
   b. 对于 Envoy image (Envoy 映像)，使用自动填充的 Envoy 容器映像，即 111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-prod。
   c. 对于 Mesh name (网格名称)，选择要使用的 App Mesh 服务网格。必须已创建才会显示。有关更多信息，请参阅 AWS App Mesh 用户指南 中的服务网格。
   d. 对于 Virtual node name (虚拟节点名称)，选择要使用的 App Mesh 虚拟节点。必须已创建才会显示。有关更多信息，请参阅 AWS App Mesh 用户指南 中的虚拟节点。
   e. 对于 Virtual node port (虚拟节点端口)，将使用虚拟节点上设置的侦听器端口进行预先填充。
   f. 选择 Apply (应用)、Confirm (确认)。这将为任务定义创建一个新的 Envoy 代理容器以及支持它的设置。然后，它将预先填充下一步中的 App Mesh 代理配置设置。
8. （可选）对于 Proxy Configuration (代理配置)，请验证所有预填充值。有关这些字段的更多信息，请参阅Proxy Configuration (p. 547)。
9. （可选）要为您的任务定义数据卷，请选择 Add volume (添加卷)。有关更多信息，请参阅在任务中使用数据卷 (p. 83)。
   - 对于 Name (名称)，键入卷名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
10. 在 Tags (标签) 部分中，为要与任务定义关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
11. 选择 Create。

EC2 launch type

使用 EC2 启动类型兼容性模板

如果选择 EC2，请完成以下步骤：

1. (可选) 如果您有任务定义的 JSON 表示形式，请完成以下步骤：
   a. 在 Configure task and container definitions 页面上，滚动到页面底部并选择 Configure via JSON。

b. 将您的任务定义 JSON 粘贴到文本区域并选择 Save。

c. 验证您的信息并选择 Create。

滚动到页面底部，选择 Configure via JSON (通过 JSON 配置)。

2. 在 Task Definition Name (任务定义名称) 中，为任务定义键入名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。

3. （可选）对于 Task Role (任务角色)，选择 IAM 角色，该角色可提供权限，允许任务中的容器代表您调用 AWS API。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

   对于使用 EC2 启动类型的任务，这些权限通常由 Amazon ECS 容器实例 IAM 角色授予。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

   Note

   此处仅显示具有 Amazon EC2 Container Service Task Role 信任关系的角色。有关如何为任务创建 IAM 角色的更多信息，请参阅 为任务创建 IAM 角色和策略 (p. 420)。

4. (可选) 对于 Network Mode，选择 Docker 网络模式，供任务中的容器使用。可用网络模式对应于 Docker 运行参考中的网络设置中描述的网络模式。

   默认 Docker 网络模式为 `bridge`。如果将网络模式设置为 `none`，则无法在容器定义中指定端口映射，并且任务的容器没有外部连接性。如果网络模式为 `awsvpc`，则任务会分配一个弹性网络接口。`host` 和 `awsvpc` 网络模式为容器提供最高联网性能，因为容器使用的是 Amazon EC2 网络堆栈而不是由 `bridge` 模式提供的虚拟化网络堆栈；不过，公开的容器端口将直接映射到相应的主机端口，因此，无法利用动态主机端口映射或在单个容器实例上运行同一任务的多个实例化（如果使用了端口映射）。

5. （可选）对于 Task execution role (任务执行角色)，选择 IAM 角色，该角色可提供权限，允许任务中的容器代表您调用 AWS API。

   对于使用 EC2 启动类型的任务，这些权限通常由 Amazon ECS 容器实例 IAM 角色授予，该角色在前面指定为 Task Role (任务角色)。无需指定任务执行角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色 (p. 406)。

6. (可选) 对于 Task size，选择 Task memory (GB) 和 Task CPU (vCPU) 的值。受支持的 `Task CPU (vCPU)` 值介于 128 个 CPU 单元 (0.125 个 vCPU) 和 10240 个 CPU 单元 (10 个 vCPU) 之间。

   Note

   Windows 容器将忽略任务级 CPU 和内存参数。我们建议为 Windows 容器指定容器级资源。

7. 对于您的任务定义中的每个容器，请完成以下步骤。

   a. 选择 Add container。

   b. 填写要在您的容器定义中使用的每个必填字段和任何可选字段（Advanced container configuration 菜单提供了更多的容器定义参数）。有关更多信息，请参阅任务定义参数 (p. 52)。

   c. 选择 Add，将容器添加到任务定义。

8. (可选) 对于 Constraint，请定义从这个任务定义创建的任务放置到您的集群中的方式。对于使用 EC2 启动类型的任务，则可以使用约束来根据可用区、实例类型或自定义属性来放置任务。有关更多信息，请参阅Amazon ECS 任务放置约束 (p. 257)。

9. （可选）对于 Service Integration (服务集成)，要配置 App Mesh 集成的参数，请选择 Enable App Mesh integration (启用 App Mesh 集成)，然后执行以下操作：

   a. 对于 Application container name (应用程序容器名称)，选择要用于 App Mesh 应用程序的容器名称。必须已在任务定义中定义此容器。

   b. 对于 Envoy image (Envoy 映像)，使用自动填充的 Envoy 容器映像，即 111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-prod。

c. 对于 Mesh name (网格名称)，选择要使用的 App Mesh 服务网格。必须已创建才会显示。有关更多信息，请参阅 AWS App Mesh 用户指南 中的服务网格。
d. 对于 Virtual node name (虚拟节点名称)，选择要使用的 App Mesh 虚拟节点。必须已创建才会显示。有关更多信息，请参阅 AWS App Mesh 用户指南 中的虚拟节点。
e. 对于 Virtual node port (虚拟节点端口)，将使用虚拟节点上设置的侦听器端口进行预先填充。
f. 选择 Apply (应用)、Confirm (确认)。这将为任务定义创建一个新的 Envoy 代理容器以及支持它的设置。然后，它将预先填充下一步中的 App Mesh 代理配置设置。

10. （可选）对于 Proxy Configuration (代理配置)，请验证所有预填充值。有关这些字段的更多信息，请参阅Proxy Configuration (p. 547)。
11. （可选）要为您的任务定义数据卷，请选择 Add volume。您可以创建一个绑定挂载或 Docker 卷。有关更多信息，请参阅在任务中使用数据卷 (p. 83)。

    a. 对于 Name (名称)，键入卷名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
    b. （可选）要创建绑定挂载卷，对于 Source path (源路径)，键入要提供给容器的主机容器实例上的路径。如果您将此字段保留为空，Docker 守护程序将为您分配一个主机路径。如果您指定源路径，数据卷将在主机容器实例上的指定位置保留，除非您手动将其删除。如果主机容器实例上不存在源路径，则 Docker 守护程序会创建它。如果该位置不存在，则源路径文件夹的内容将导出至容器。
    c. 要创建 Docker 卷，选择 Specify a volume driver (指定卷驱动程序)。

        i. 对于 Driver (驱动程序)，选择要使用的 Docker 卷驱动程序。驱动程序值必须与 Docker 提供的驱动程序名称相符。对您的容器实例使用 `docker plugin ls` 来检索驱动程序名称。
        ii. 对于 Scope (范围)，选择确定 Docker 卷的生命周期的选项。当任务开始时，将自动预置范围限定为 `task` 的 Docker 卷；而当任务停止时销毁此卷。任务停止后，范围限定为 `shared` 的 Docker 卷将持续存在。
        iii. 选择 Enable auto-provisioning (启用自动预置) 以创建此 Docker 卷（如果它尚不存在）。此选项仅适用于指定 `shared` 范围的卷。
        iv. 对于 Driver options (驱动程序选项)，指定要使用的驱动程序特定的密钥值。
        v. 对于 Volume labels (卷标签)，指定要添加到 Docker 卷的自定义元数据。

12. 在 Tags (标签) 部分中，为要与任务定义关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
13. 选择 Create (创建)。

任务定义模板

空任务定义模板如下所示。您可以使用此模板创建任务定义，这些任务定义随后可粘贴到控制台 JSON 输入区域或保存到文件并与 AWS CLI `--cli-input-json` 选项结合使用。有关更多信息，请参阅任务定义参数 (p. 52)。

```
{
    "family": "",
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "none",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {
                "credentialsParameter": ""
            },
            "cpu": 0,
```

```
            "memory": 0,
            "memoryReservation": 0,
            "links": [
                ""
            ],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                ""
            ],
            "command": [
                ""
            ],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
                    "sourceContainer": "",
                    "readOnly": true
                }
            ],
            "linuxParameters": {
                "capabilities": {
                    "add": [
                        ""
                    ],
                    "drop": [
                        ""
                    ]
                },
                "devices": [
                    {
                        "hostPath": "",
                        "containerPath": "",
                        "permissions": [
                            "mknod"
                        ]
                    }
                ],
                "initProcessEnabled": true,
                "sharedMemorySize": 0,
                "tmpfs": [
                    {
                        "containerPath": "",
                        "size": 0,
                        "mountOptions": [
                            ""
                        ]
                    }
```

```
            ]
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "SUCCESS"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
        "user": "",
        "workingDirectory": "",
        "disableNetworking": true,
        "privileged": true,
        "readonlyRootFilesystem": true,
        "dnsServers": [
            ""
        ],
        "dnsSearchDomains": [
            ""
        ],
        "extraHosts": [
            {
                "hostname": "",
                "ipAddress": ""
            }
        ],
        "dockerSecurityOptions": [
            ""
        ],
        "interactive": true,
        "pseudoTerminal": true,
        "dockerLabels": {
            "KeyName": ""
        },
        "ulimits": [
            {
                "name": "rss",
                "softLimit": 0,
                "hardLimit": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "syslog",
            "options": {
                "KeyName": ""
            }
        },
        "healthCheck": {
            "command": [
                ""
            ],
            "interval": 0,
            "timeout": 0,
            "retries": 0,
            "startPeriod": 0
        },
        "systemControls": [
            {
```

```
                    "namespace": "",
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "GPU"
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {
                "sourcePath": ""
            },
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {
                    "KeyName": ""
                },
                "labels": {
                    "KeyName": ""
                }
            }
        }
    ],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": ""
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "pidMode": "host",
    "ipcMode": "host",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    }
}
```

您可以使用以下 AWS CLI 命令生成此任务定义模板：

```
aws ecs register-task-definition --generate-cli-skeleton
```

任务定义参数

任务定义包括以下各部分：任务系列、IAM 任务角色、网络模式、容器定义、卷、任务放置约束和启动类型。系列和容器定义在任务定义中是必需的，而任务角色、网络模式、卷、任务放置约束和启动类型是可选的。

以下是每个任务定义参数的更详细说明。

系列

`family`

类型：字符串

必需：是

当您注册任务定义时，将为其提供一个系列，与任务定义的多个版本的名称类似，它是使用修订号指定的。注册到某个特定系列的第一个任务定义将获得修订 1，而随后注册的任何任务定义将获得后续修订号。

任务角色

`taskRoleArn`

类型：字符串

必需：否

注册任务定义时，可以为 IAM 角色提供任务角色，此角色可使任务中的容器有权代表您调用其关联策略中指定的 AWS API。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

Windows 上任务的 IAM 角色需要在您启动Amazon ECS-optimized Windows AMI 时设置 `-EnableTaskIAMRole` 选项。此外，您的容器必须运行一些配置代码才能利用此功能。有关更多信息，请参阅 任务的 Windows IAM 角色 (p. 608)。

任务执行角色

`executionRoleArn`

类型：字符串

必需：否

注册任务定义时，您可以提供任务执行角色，允许任务中的容器代表您提取容器映像并将容器日志发送到 CloudWatch。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

网络模式

`networkMode`

类型：字符串

必需：否

在任务中用于容器的 Docker 联网模式有效值为 `none`、`bridge`、`awsvpc` 和 `host`。默认 Docker 网络模式为 `bridge`。

如果网络模式设置为 `none`，则任务的容器没有外部连接性，且无法在容器定义中指定端口映射。

如果网络模式为 `bridge`，该任务利用在每个容器实例内运行的 Docker 的内置虚拟网络。

如果网络模式为 `host`，则任务将绕过 Docker 的内置虚拟网络，并将容器端口直接映射到 EC2 实例的网络接口。在此模式下，当使用端口映射时，无法在单个容器实例上运行同一任务的多个实例化。

如果网络模式为 `awsvpc`，则将为任务分配弹性网络接口，且在创建服务或运行具有任务定义的任务时必须指定 `NetworkConfiguration`。有关更多信息，请参阅awsvpc 网络模式的任务联网 (p. 95)。目前，只有Amazon ECS-optimized AMI、包含 `ecs-init` 程序包的其他 Amazon Linux 变体或 AWS Fargate 基础设施才支持 `awsvpc` 网络模式。

`host` 和 `awsvpc` 网络模式为容器提供最高的联网性能，因为它们使用 Amazon EC2 网络堆栈，而不是 `bridge` 模式提供的虚拟化网络堆栈。使用 `host` 和 `awsvpc` 网络模式，公开的容器端口将直接映射到相应的主机端口 (用于 `host` 网络模式) 或附加的弹性网络接口端口 (用于 `awsvpc` 网络模式)，因此您无法利用动态主机端口映射。

Docker for Windows 使用的网络模式（也称为 `NAT`）不同于 Docker for Linux。在将任务定义注册到 Windows 容器时，您不得指定网络模式。如果使用 AWS 管理控制台将任务定义注册到 Windows 容器，必须选择 `default` 网络模式。

如果使用 Fargate 启动类型，则需要 `awsvpc` 网络模式。如果使用 EC2 启动类型，则允许的网络模式取决于底层 EC2 实例的操作系统。对于 Linux，可以使用任何网络模式。对于 Windows，仅允许上面所述的 `NAT` 模式。

容器定义

当您注册任务定义时，必须指定将传递给容器实例上的 Docker 守护程序的容器定义的列表。容器定义中允许以下参数。

主题

- 标准容器定义参数 (p. 53)
- 高级容器定义参数 (p. 57)
- 其他容器定义参数 (p. 68)

标准容器定义参数

以下任务定义参数是必需的参数或在大多数容器定义中使用。

主题

- 名称 (p. 53)
- 映像 (p. 54)
- 内存 (p. 54)
- 端口映射 (p. 55)

名称

`name`

类型：字符串

必需：是

容器的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。如果您正在任务定义中将多个容器链接在一起，则可在一个容器的 `links` 中输入另一个容器的 `name` 以连接容器。

映像

`image`

类型：字符串

必需：是

用于启动容器的映像。此字符串将直接传递给 Docker 守护程序。默认情况下，Docker Hub 注册表中的映像可用。您也可以使用 *repository-url*/*image*:*tag* 或 *repository-url*/*image*@*digest* 指定其他存储库。允许最多 255 个字母 (大写和小写字母)、数字、连字符、下划线、冒号、句点、正斜杠和井号。此参数将映射到 Docker Remote API 的创建容器部分中的 `Image` 以及 docker run 的 `IMAGE` 参数。

- 在新任务启动时，Amazon ECS 容器代理会提取最新版本的指定映像和标签以供容器使用。但是，存储库映像的后续更新不会传播到已在运行的任务。
- 支持私有注册表中的映像。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。
- Amazon ECR 存储库中的映像可通过使用完整的 `registry/repository:tag` 或 `registry/repository@digest` 命名约定来指定。例如，*aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest* 或 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*@*sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE*。
- Docker Hub 上的官方存储库中的映像使用一个名称（例如，`ubuntu` 或 `mongo`）。
- Docker Hub 上其他存储库中的映像通过组织名称（例如，`amazon/amazon-ecs-agent`）进行限定。
- 其他在线存储库中的映像由域名 (例如，`quay.io/assemblyline/ubuntu`) 进行进一步限定。

内存

`memory`

类型：整数

必需：否

要提供给容器的内存量（以 MiB 为单位）。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。为任务中的所有容器预留的内存总量必须低于任务 `memory` 值（如果指定了一个）。此参数将映射到 Docker Remote API 的创建容器部分中的 `Memory` 以及 docker run 的 `--memory` 选项。

如果您的容器是使用 Fargate 启动类型的任务的一部分，则此字段是可选的。

对于属于使用 EC2 启动类型的任务的容器，您必须在容器定义中为 `memory` 和/或 `memoryReservation` 指定非零整数。如果指定两者，则 `memory` 必须大于 `memoryReservation`。如果指定 `memoryReservation`，则将从容器所在的容器实例的可用内存资源中减去该值；否则将使用 `memory` 的值。

Docker 守护程序将为容器预留最少 4 MiB 的内存，因此，您不应为容器指定 4 MiB 以下的内存。

> **Note**
>
> 如果您尝试通过为任务提供尽可能多的用于特定实例类型的内存来最大程度地利用资源，请参阅容器实例内存管理 (p. 190)。

`memoryReservation`

类型：整数

必需：否

要为容器预留的内存量的软限制（以 MiB 为单位）。当系统内存处于争用状态时，Docker 会尝试将容器内存保持在此软限制内；不过，您的容器可以在需要时使用更多内存，最大量可达 `memory` 参数指定的硬限制（如果适用），或容器实例上的所有可用内存（以先达到者为准）。此参数将映射到 Docker Remote API 的创建容器部分中的 `MemoryReservation` 以及 docker run 的 `--memory-reservation` 选项。

您必须为容器定义中的 `memory` 和/或 `memoryReservation` 指定一个非零整数。如果指定两者，则 `memory` 必须大于 `memoryReservation`。如果指定 `memoryReservation`，则将从容器所在的容器实例的可用内存资源中减去该值；否则将使用 `memory` 的值。

例如，如果您的容器通常使用 128 MiB 内存，但有时会在短时间内会迸发至 256 MiB 内存，则您可以设置 128 MiB 的 `memoryReservation` 以及 300 MiB 的 `memory` 硬限制。此配置不仅允许容器从容器实例上的剩余资源中预留 128 MiB 内存，而且允许容器在需要时使用更多内存资源。

Docker 守护程序将为容器预留最少 4 MiB 的内存，因此，您不应为容器指定 4 MiB 以下的内存。

端口映射

`portMappings`

类型：对象数组

必需：否

端口映射可让容器访问主机容器实例上的端口以发送或接收流量。

对于使用 `awsvpc` 网络模式的任务定义，您只应指定 `containerPort`。`hostPort` 可以留空或值必须与 `containerPort` 相同。

Windows 上的端口映射使用 `NetNAT` 网关地址而非 `localhost`。没有针对 Windows 上的端口映射的回环，因此，您无法从主机自身访问容器的映射端口。

此参数将映射到 Docker Remote API 的创建容器部分中的 `PortBindings` 以及 docker run 的 `--publish` 选项。如果将任务定义的网络模式设置为 `host`，则主机端口必须是未定义的或者必须与端口映射中的容器端口匹配。

> **Note**
>
> 在任务达到 `RUNNING` 状态之后，手动和自动主机及容器端口分配在以下位置可见：
>
> - 控制台：所选任务的容器说明的网络绑定部分。
> - AWS CLI：describe-tasks 命令输出的 `networkBindings` 部分。
> - API：`DescribeTasks` 响应。

`containerPort`

类型：整数

必需：是，当使用 `portMappings` 时

绑定到用户指定的或自动分配的主机端口的容器上的端口号。

如果在具有 Fargate 的任务中使用容器，公开的端口应使用 `containerPort` 指定。

如果在具有 EC2 启动类型的任务中使用容器，且指定了容器端口而不是主机端口，则您的容器将自动接收临时端口范围中的主机端口。有关更多信息，请参阅`hostPort`。通过此方式自动分配的端口映射不计入容器实例的 100 个预留端口限制。

> **Important**
>
> 您不能为多个协议公开相同的容器端口。如果您尝试这样做，则会返回错误。

`hostPort`

类型：整数

必需：否

要为您的容器预留的容器实例上的端口号。

如果在具有 Fargate 启动类型的任务中使用容器，则 `hostPort` 可以留空或为与 `containerPort` 相同的值。

如果在具有 EC2 启动类型的任务中使用容器，则您可以为容器端口映射指定非预留主机端口（这称为静态主机端口映射），也可以在指定 `containerPort` 时省略 `hostPort`（或将它设置为 `0`），您的容器将自动接收容器实例操作系统和 Docker 版本的临时端口范围中的端口（这称为动态主机端口映射）。

默认临时端口范围为 `49153-65535`，此范围用于 1.6.0 版本之前的 Docker。对于 Docker 版本 1.6.0 及更高版本，Docker 守护程序将尝试读取 `/proc/sys/net/ipv4/ip_local_port_range` 中的临时端口范围（此范围在最新Amazon ECS-optimized AMI 上为 32768–61000）；如果此内核参数不可用，则将使用默认临时端口范围。不要尝试指定位于临时端口范围中的主机端口，因为这些端口保留供自动分配。通常，低于 32768 的端口位于临时端口范围之外。

默认预留端口为适用于 SSH 的端口 `22`、Docker 端口 `2375` 和 `2376` 以及 Amazon ECS 容器代理端口 `51678-51680`。针对正在运行的任务的之前由用户指定的任何主机端口在任务运行时也将预留（在任务停止后，将发布主机端口）。当前预留端口将显示在 describe-container-instances 输出的 `remainingResources` 中，并且一个容器实例一次最多可拥有 100 个预留端口，包括默认预留端口。自动分配的端口不计入 100 个预留端口限制。

`protocol`

类型：字符串

必需：否

用于端口映射的协议。有效值为 `tcp` 和 `udp`。默认为 `tcp`。

> **Important**
>
> UDP 支持仅适用于使用 Amazon ECS 容器代理的 1.2.0 版（例如 `amzn-ami-2015.03.c-amazon-ecs-optimized` AMI）或更高版本或者使用已更新至版本 1.3.0 或更高版本的容器代理启动的容器实例。要将您的容器代理更新至最新版本，请参阅更新 Amazon ECS 容器代理 (p. 211)。

如果您指定的是主机端口，请使用以下语法：

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
]
```

如果您需要自动分配的主机端口，请使用以下语法：

```
"portMappings": [
    {
        "containerPort": integer
    }
    ...
]
```

高级容器定义参数

以下高级容器定义参数提供用于在 Amazon ECS 容器实例上启动容器的 docker run 命令的扩展功能。

主题

- 运行状况检查 (p. 57)
- 环境 (p. 58)
- 网络设置 (p. 61)
- 存储和日志记录 (p. 63)
- 安全性 (p. 66)
- 资源限制 (p. 67)
- Docker 标签 (p. 68)

运行状况检查

`healthCheck`

容器的运行状况检查命令和关联的配置参数。该参数映射到 Docker Remote API 的 创建容器 部分中的 `HealthCheck` 以及 docker run 的 `HEALTHCHECK` 参数。

> **Note**
>
> Amazon ECS 容器代理仅监控和报告在任务定义中指定的运行状况检查。Amazon ECS 不监控嵌入到容器映像中和未在容器定义中指定的 Docker 运行状况检查。在容器定义中指定的运行状况检查参数覆盖容器映像中存在的任意 Docker 运行状况检查。

任务运行状况由任务的 `healthStatus` 报告，它由任务中的主要容器的运行状况确定。如果任务中的所有主要容器报告为 `HEALTHY`，则任务状态也报告为 `HEALTHY`。如果任务中的任意主要容器报告为 `UNHEALTHY` 或 `UNKNOWN`，则任务状态也报告为相应报告为 `UNHEALTHY` 或 `UNKNOWN`。如果服务的任务报告为不正常，则从服务中删除该任务并替换。

以下是有关容器运行状况检查支持的注意事项：

- 容器运行状况检查需要 Amazon ECS 容器代理版本 1.17.0 或更高版本。有关更多信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。
- 如果使用平台版本 1.1.0 或更高版本，则 Fargate 任务支持容器运行状况检查。有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。
- 如果任务属于配置为使用 传统负载均衡器 的服务的一部分，则此类任务不支持容器运行状况检查。

`command`

一个表示容器运行的命令的字符串数组，用于确定其运行状况是否正常。字符串数组可以以 `CMD` 开头以直接执行命令参数，或者以 `CMD-SHELL` 开头以使用容器的默认 Shell 来运行命令。如果两者都未指定，将默认使用 `CMD`。

在 AWS 管理控制台中注册任务定义时，请使用命令的逗号分隔列表，这些命令在创建任务定义后将自动转换为字符串。运行状况检查的示例输入可以是：

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

在使用 AWS 管理控制台 JSON 面板、AWS CLI 或 API 注册任务定义时，您应将命令列表包含在括号中。运行状况检查的示例输入可以是：

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

退出代码 0 表示成功，非零退出代码表示失败。有关更多信息，请参阅 Docker Remote API 的创建容器部分中的 `HealthCheck`。

`interval`

执行每次运行状况检查间隔的时间，以秒为单位。您可以指定 5 到 300 秒之间的值。默认值为 30 秒。

`timeout`

等待运行状况检查成功执行的时间长度，以秒为单位，超过该时间则视为失败。您可以指定 2 到 60 秒之间的值。默认值为 5 秒。

`retries`

重试失败的运行状况检查的次数，超过该次数将容器视为不正常。您可以指定 1 到 10 秒之间的重试次数。默认值为三次重试。

`startPeriod`

可选的宽限期，这让容器有时间来引导，不将失败的运行状况检查计数计入最大重试次数中。您可以指定 0 到 300 秒之间的值。默认情况下禁用 `startPeriod`。

环境

`cpu`

类型：整数

必需：否

Amazon ECS 容器代理将为容器预留的 `cpu` 单元的数量。此参数将映射到 Docker Remote API 的创建容器部分中的 `CpuShares` 以及 docker run 的 `--cpu-shares` 选项。

此字段对于使用 Fargate 启动类型的任务为可选，而且唯一的要求是任务内为所有容器保留的 CPU 总量低于任务级 `cpu` 值。

> **Note**
>
> 您可以通过将 Amazon EC2 实例详细信息页面上列出的每个 Amazon EC2 实例类型的 vCPU 数乘以 1024 来确定该实例类型可用的 CPU 单元数。

Linux 容器会按照与其分配的量相同的比例，与容器实例上的其他容器共享未分配的 CPU 单元。例如，如果您在单核实例类型上运行一个单容器任务，同时为该容器指定 512 个 CPU 单元，而且这是在容器实例上运行的唯一任务，则该容器可在任何给定间使用完整的 1024 个 CPU 单元份额。但是，如果您在该容器实例上启动了同一任务的另一个副本，则保证为每个任务提供最少 512 个 CPU 单元（如果需要），而且每个容器的 CPU 使用量会上浮到更高的值（如果另一容器未在使用它），但如果两个任务都始终处于完全活动状态，则它们将限于 512 个 CPU 单元。

在 Linux 容器实例上，容器实例上的 Docker 守护程序使用 CPU 值来计算正在运行的容器的相对 CPU 共享比例。有关更多信息，请参阅 Docker 文档中的 CPU 共享约束。Linux 内核允许的最小有效 CPU 共享值为 2。不过，CPU 参数不是必需的，您可以在容器定义中使用小于 2 的 CPU 值。对于小于 2 的 CPU 值（包括 null），此行为因您的 Amazon ECS 容器代理版本而异：

- 代理版本 <= 1.1.0：Null 和零 CPU 值将作为 0 传递给 Docker，然后 Docker 将其转换为 1,024 个 CPU 份额。CPU 值 1 将作为 1 传递给 Docker，然后 Linux 内核将其转换为 2 个 CPU 份额。
- 代理版本 >= 1.2.0：Null、零和 CPU 值 1 将作为 2 个 CPU 份额传递给 Docker。

在 Windows 容器实例上，此 CPU 限制将作为绝对限制或配额强制实施。Windows 容器只能访问任务定义中所述的指定的 CPU 量。

`gpu`

类型：ResourceRequirement 对象

必需：否

Amazon ECS 容器代理将为容器预留的物理 `GPUs` 的数量。为某个任务中的所有容器预留的 GPU 的数量不应超过该任务在其上启动的容器实例的可用 GPU 的数量。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。

> **Note**
>
> 使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

`essential`

类型：布尔值

必需：否

如果将容器的 `essential` 参数标记为 `true`，并且该容器出于任何原因发生故障或停止，则属于此任务的所有其他容器将停止。如果将容器的 `essential` 参数标记为 `false`，则容器发生故障不会影响任务中的剩余容器。如果省略此参数，则假定容器是主要容器。

所有任务都必须具有至少一个主要容器。如果您有一个由多个容器组成的应用程序，则应将用于相同目的的容器分成多个组件，然后将不同的组件分为多个任务定义。有关更多信息，请参阅应用程序架构 (p. 43)。

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> Amazon ECS 容器代理的早期版本无法正确处理 `entryPoint` 参数。如果您在使用 `entryPoint` 时遇到问题，请更新您的容器代理或改为输入命令和参数作为 `command` 数组项。

类型：字符串数组

必需：否

传递给容器的入口点。此参数将映射到 Docker Remote API 的创建容器部分中的 `Entrypoint` 以及 docker run 的 `--entrypoint` 选项。有关 Docker `ENTRYPOINT` 参数的更多信息，请转到 https://docs.docker.com/engine/reference/builder/#entrypoint。

```
"entryPoint": ["string", ...]
```

`command`

类型：字符串数组

必需：否

传递给容器的命令。此参数将映射到 Docker Remote API 的创建容器部分中的 `Cmd` 以及 docker run 的 `COMMAND` 参数。有关 Docker `CMD` 参数的更多信息，请转到 https://docs.docker.com/engine/reference/builder/#cmd。如果有多个参数，则每个参数应该是数组中的分隔字符串。

```
"command": ["string", ...]
```

`workingDirectory`

类型：字符串

必需：否

容器中用于运行命令的工作目录。此参数将映射到 Docker Remote API 的创建容器部分中的 `WorkingDir` 以及 docker run 的 `--workdir` 选项。

```
"workingDirectory": "string"
```

`environment`

类型：对象数组

必需：否

要传递给容器的环境变量。此参数将映射到 Docker Remote API 的创建容器部分中的 `Env` 以及 docker run 的 `--env` 选项。

> **Important**
>
> 建议不要对敏感信息 (如凭证数据) 使用纯文本环境变量。

`name`

类型：字符串

必需：是，当使用 `environment` 时

环境变量的名称。

`value`

类型：字符串

必需：是，当使用 `environment` 时

环境变量的值。

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

`secrets`

类型：对象数组

必需：否

一个表示用于对您的容器开放的密钥的对象。有关更多信息，请参阅指定敏感数据 (p. 107)。

`name`

类型：字符串

必需：是

要在容器上设置为环境变量的值。

`valueFrom`

类型：字符串

必需：是

要向容器公开的密钥。支持的值为 AWS Secrets Manager 密钥的完整 ARN 或 AWS Systems Manager Parameter Store 中的参数的完整 ARN。

Note

如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用密钥的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

```
"secrets": [
    {
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

网络设置

`disableNetworking`

类型：布尔值

必需：否

当此参数设置为 true 时，容器内禁用联网。此参数将映射到 Docker Remote API 的创建容器部分中的 `NetworkDisabled`。

Note

使用 `awsvpc` 网络模式的 Windows 容器或任务不支持此参数。

```
"disableNetworking": true|false
```

`links`

类型：字符串数组

必需：否

`link` 参数允许容器相互进行通信，而无需端口映射。只有在任务定义的网络模式设置为 `bridge` 才支持。`name:internalName` 构造类似于 Docker 链接中的 `name:alias`。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。 有关链接 Docker 容器的更多信息，请转到 https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/。此参数将映射到 Docker Remote API 的创建容器部分中的 `Links` 以及 docker run 的 `--link` 选项。

Note

使用 `awsvpc` 网络模式的 Windows 容器或任务不支持此参数。

Important

在单个容器实例上并置的容器也许能够相互进行通信，而无需链接或主机端口映射。可使用安全组和 VPC 设置在容器实例上实现网络隔离。

```
"links": ["name:internalName", ...]
```

`hostname`

类型：字符串

必需：否

要对您的容器使用的主机名。此参数将映射到 Docker Remote API 的创建容器部分中的 `Hostname` 以及 docker run 的 `--hostname` 选项。

> **Note**
>
> 如果使用 `awsvpc` 网络模式，则不支持 `hostname` 参数。

```
"hostname": "string"
```

`dnsServers`

类型：字符串数组

必需：否

提供给容器的 DNS 服务器的列表。此参数将映射到 Docker Remote API 的创建容器部分中的 `Dns` 以及 docker run 的 `--dns` 选项。

> **Note**
>
> 使用 `awsvpc` 网络模式的 Windows 容器或任务不支持此参数。

```
"dnsServers": ["string", ...]
```

`dnsSearchDomains`

类型：字符串数组

必需：否

模式：^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$

提供给容器的 DNS 搜索域的列表。此参数将映射到 Docker Remote API 的创建容器部分中的 `DnsSearch` 以及 docker run 的 `--dns-search` 选项。

> **Note**
>
> 使用 `awsvpc` 网络模式的 Windows 容器或任务不支持此参数。

```
"dnsSearchDomains": ["string", ...]
```

`extraHosts`

类型：对象数组

必需：否

要追加到容器上的 `/etc/hosts` 文件的主机名和 IP 地址映射的列表。

此参数将映射到 Docker Remote API 的创建容器部分中的 `ExtraHosts` 以及 docker run 的 `--add-host` 选项。

> **Note**
>
> 使用 `awsvpc` 网络模式的 Windows 容器或任务不支持此参数。

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
```

```
    }
    ...
  ]
```

`hostname`

类型：字符串

必需：是，当使用 `extraHosts` 时

要用于 `/etc/hosts` 条目中的主机名。

`ipAddress`

类型：字符串

必需：是，当使用 `extraHosts` 时

要用于 `/etc/hosts` 条目中的 IP 地址。

存储和日志记录

`readonlyRootFilesystem`

类型：布尔值

必需：否

当此参数为 true 时，将对此容器提供对其根文件系统的只读访问权。此参数将映射到 Docker Remote API 的创建容器部分中的 `ReadonlyRootfs` 以及 docker run 的 `--read-only` 选项。

> **Note**
>
> Windows 容器不支持此参数。

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

Type: Object

Required: No

The mount points for data volumes in your container.

This parameter maps to `Volumes` in the Create a container section of the Docker Remote API and the `--volume` option to docker run.

Windows containers can mount whole directories on the same drive as `$env:ProgramData`. Windows containers cannot mount directories on a different drive, and mount point cannot be across drives.

`sourceVolume`

Type: String

Required: Yes, when `mountPoints` are used

The name of the volume to mount.

`containerPath`

Type: String

Required: Yes, when `mountPoints` are used

The path on the container to mount the volume at.

`readOnly`

Type: Boolean

Required: No

If this value is `true`, the container has read-only access to the volume. If this value is `false`, then the container can write to the volume. The default value is `false`.

`volumesFrom`

类型：对象数组

必需：否

要从其他容器挂载的数据卷。此参数将映射到 Docker Remote API 的创建容器部分中的 `VolumesFrom` 以及 docker run 的 `--volumes-from` 选项。

`sourceContainer`

类型：字符串

必需：是，当使用 `volumesFrom` 时

要从其中挂载卷的容器的名称。

`readOnly`

类型：布尔值

必需：否

如果此值为 `true`，则容器具有对卷的只读访问权。如果此值为 `false`，则容器可对卷进行写入。默认值为 `false`。

```
"volumesFrom": [
                {
                  "sourceContainer": "string",
                  "readOnly": true|false
                }
              ]
```

`logConfiguration`

类型：LogConfiguration 对象

必需：否

容器的日志配置规范。

有关在任务定义中使用 `awslogs` 日志驱动程序以将容器日志发送到 CloudWatch Logs 的更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。

有关使用日志配置的示例任务定义，请参阅示例任务定义 (p. 112)。

此参数将映射到 Docker Remote API 的创建容器部分中的 `LogConfig` 以及 `docker run` 的 `--log-driver` 选项。默认情况下，容器使用 Docker 守护程序所用的同一日志记录驱动程序；但容器可能通过在容器定义中使用此参数指定日志驱动程序来使用不同于 Docker 守护程序的日志记录驱动程序。要对容器使用不同的日志记录驱动程序，必须在容器实例上正确配置日志系统（或者在不同的日志服务器上使用远程日志记录选项）。有关其他支持的日志驱动程序选项的更多信息，请参阅 Docker 文档中的配置日志记录驱动程序。

指定容器的日志配置时，应注意以下事项：

- Amazon ECS 目前支持提供给 Docker 守护程序的一小部分日志记录驱动程序（如以下有效值所示）。可能会在 Amazon ECS 容器代理的未来版本中提供其他日志驱动程序。
- 此参数要求您容器实例上的 Docker Remote API 版本为 1.18 或更高。
- 对于使用 EC2 启动类型的任务，在容器实例上运行的 Amazon ECS 容器代理必须先将该实例上的可用日志记录驱动程序注册到 `ECS_AVAILABLE_LOGGING_DRIVERS` 环境变量，然后该实例上的容器才能使用这些日志配置选项。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。
- 对于使用 Fargate 启动类型的任务，因为您无权访问托管您的任务的底层基础设施，所以必须在此任务之外安装任何所需的其他软件。例如，Fluentd 输出聚合函数或运行 Logstash 以将 Gelf 日志发送到的远程主机。

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","logentries","splunk","syslog",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

`logDriver`

类型：字符串

有效值：`"awslogs","fluentd","gelf","json-file","journald","logentries","splunk","syslog"`

必需：是，当使用 `logConfiguration` 时

要用于容器的日志驱动程序。之前列出的有效值是默认情况下 Amazon ECS 容器代理可与之通信的日志驱动程序。

对于使用 Fargate 启动类型的任务，受支持的日志驱动程序为 `awslogs` 和 `splunk`。

对于使用 EC2 启动类型的任务，受支持的日志驱动程序为 `awslogs`、`fluentd`、`gelf`、`json-file`、`journald`、`logentries`、`splunk` 和 `syslog`。

> **Note**
>
> 如果您有未列出的自定义驱动程序，则可使 GitHub 上提供的 Amazon ECS 容器代理项目分叉，并对其进行自定义以便与该驱动程序结合使用。我们鼓励您针对要包含的更改提交提取请求。但是，我们当前不支持运行此软件的修改后副本。

此参数要求您容器实例上的 Docker Remote API 版本为 1.18 或更高。

`options`

类型：字符串到字符串映射

必需：否

要发送到日志驱动程序的配置选项。

此参数要求您容器实例上的 Docker Remote API 版本为 1.19 或更高。

`secretOptions`

类型：对象数组

必需：否

一个对象，表示要传递到日志配置的密钥。有关更多信息，请参阅指定敏感数据 (p. 107)。

`name`

类型：字符串

必需：是

要在容器上设置为环境变量的值。

`valueFrom`

类型：字符串

必需：是

要向容器的日志配置公开的密钥。

```
"logConfiguration": {
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
}
```

安全性

`privileged`

类型：布尔值

必需：否

当此参数为 true 时，将对此容器提供对主机容器实例的提升的特权（类似于 `root` 用户）。

此参数将映射到 Docker Remote API 的创建容器部分中的 `Privileged` 以及 docker run 的 `--privileged` 选项。

> **Note**
>
> 使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

```
"privileged": true|false
```

`user`

类型：字符串

必需：否

要在容器内使用的用户名。此参数将映射到 Docker Remote API 的创建容器部分中的 `User` 以及 docker run 的 `--user` 选项。

您可以使用以下格式。如果指定 UID 或 GID，您必须将其指定为一个正整数。

- `user`

- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Windows 容器不支持此参数。

```
"user": "string"
```

`dockerSecurityOptions`

类型：字符串数组

必需：否

用于为 SELinux 和 AppArmor 多级别安全系统提供自定义标签的字符串列表。

此参数将映射到 Docker Remote API 的创建容器部分中的 `SecurityOpt` 以及 docker run 的 `--security-opt` 选项。

> **Note**
>
> 使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

```
"dockerSecurityOptions": ["string", ...]
```

> **Note**
>
> 在容器实例上运行的 Amazon ECS 容器代理必须先注册到 `ECS_SELINUX_CAPABLE=true` 或 `ECS_APPARMOR_CAPABLE=true` 环境变量，然后该实例上的容器才能使用这些安全选项。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

资源限制

`ulimits`

类型：对象数组

必需：否

要在容器中设置的 `ulimits` 的列表。此参数将映射到 Docker Remote API 的创建容器部分中的 `Ulimits` 以及 docker run 的 `--ulimit` 选项。

此参数要求您容器实例上的 Docker Remote API 版本为 1.18 或更高。

> **Note**
>
> Windows 容器不支持此参数。

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtprio"|"r
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
```

```
    ]
```

name

类型：字符串

有效值："core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"

必需：是，当使用 ulimits 时

ulimit 的 type

hardLimit

类型：整数

必需：是，当使用 ulimits 时

ulimit 类型的硬限制。

softLimit

类型：整数

必需：是，当使用 ulimits 时

ulimit 类型的软限制。

Docker 标签

dockerLabels

类型：字符串到字符串映射

必需：否

要添加到容器的标签的键值映射。此参数将映射到 Docker Remote API 的创建容器部分中的 Labels 以及 docker run 的 --label 选项。

此参数要求您容器实例上的 Docker Remote API 版本为 1.18 或更高。

```
"dockerLabels": {"string": "string"
      ...}
```

其他容器定义参数

以下容器定义参数可在 Amazon ECS 控制台中注册任务定义时使用，方法是使用 Configure via JSON (通过 JSON 配置) 选项。有关更多信息，请参阅创建任务定义 (p. 45)。

主题

- Linux 参数 (p. 69)
- 容器依赖项 (p. 71)
- 容器超时 (p. 72)
- 系统控制 (p. 73)
- 交互式 (p. 74)
- 伪终端 (p. 74)

Linux 参数

`linuxParameters`

类型：LinuxParameters 对象

必需：否

应用于容器的特定于 Linux 的选项，如 KernelCapabilities。

> **Note**
>
> Windows 容器不支持此参数。

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

类型：KernelCapabilities 对象

必需：否

容器的 Linux 功能，这些功能已在 Docker 提供的默认配置中添加或删除。有关默认功能和非默认的可用功能，请参阅 Docker 运行参考 中的运行时特权和 Linux 功能。有关这些 Linux 功能的更多详细信息，请参阅 capabilities(7) Linux 手册页面。

> **Note**
>
> 如果您使用的任务使用 Fargate 启动类型，则支持 `capabilities`，但不支持下面所述的 `add` 参数。

`add`

类型：字符串数组

有效值：`"ALL" | "AUDIT_CONTROL" | "AUDIT_READ" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

必需：否

容器的 Linux 功能，这些功能将添加到 Docker 提供的默认配置。该参数映射到 Docker Remote API 的 创建容器 部分中的 `CapAdd` 以及 docker run 的 `--cap-add` 选项。

> **Note**
>
> 如果您使用的任务使用 Fargate 启动类型，则不支持 `add` 参数。

`drop`

类型：字符串数组

有效值：`"ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"`

必需：否

容器的 Linux 功能，这些功能将从 Docker 提供的默认配置中删除。该参数映射到 Docker Remote API 的 创建容器 部分中的 `CapDrop` 以及 docker run 的 `--cap-drop` 选项。

`devices`

对容器公开的任何主机设备。该参数映射到 Docker Remote API 的 创建容器 部分中的 `Devices` 以及 docker run 的 `--device` 选项。

> **Note**
>
> 如果您使用的任务使用 Fargate 启动类型，则不支持 `devices` 参数。

类型：设备对象数组

必需：否

`hostPath`

主机容器实例上的设备路径。

类型：字符串

必需：是

`containerPath`

向主机设备公开的容器内的路径。

类型：字符串

必需：否

`permissions`

提供给设备的容器的显式权限。默认情况下，容器对设备上的 `read`、`write` 和 `mknod` 具有权限。

类型：字符串数组

有效值：`read | write | mknod`

`initProcessEnabled`

在容器内运行 `init` 进程，转发信号和获得进程。此参数会将 `--init` 选项映射到 docker run。

此参数要求您容器实例上的 Docker Remote API 版本为 1.25 或更高。

`sharedMemorySize`

`/dev/shm` 卷的大小值 (以 MiB 为单位)。此参数会将 `--shm-size` 选项映射到 docker run。

> **Note**
>
> 如果您使用的任务使用 Fargate 启动类型，则不支持 `sharedMemorySize` 参数。

类型：整数

`tmpfs`

tmpfs 挂载的容器路径、挂载选项和大小 (以 MiB 为单位)。此参数会将 `--tmpfs` 选项映射到 docker run。

> **Note**
>
> 如果您使用的任务使用 Fargate 启动类型，则不支持 `tmpfs` 参数。

类型：Tmpfs 对象数组

必需：否

`containerPath`

要装载 tmpfs 卷的绝对文件路径。

类型：字符串

必需：是

`mountOptions`

tmpfs 卷装载选项列表。

类型：字符串数组

必需：否

有效值：`"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"`

`size`

tmpfs 卷的大小 (以 MiB 为单位)。

类型：整数

必需：是

容器依赖项

`dependsOn`

类型：ContainerDependency 对象的数组

必需：否

针对容器启动和关闭定义的依赖项。一个容器可以包含多个依赖项。当针对容器启动定义依赖项时，对于容器关闭，将反转此项。有关示例，请参阅示例：容器依赖项 (p. 116)。

对于使用 EC2 启动类型的任务，容器实例需要至少 1.26.0 版的容器代理才能启用容器依赖项。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。如果您使用 Amazon ECS-optimized Amazon Linux AMI，您的实例至少需要 1.26.0-1 版的 `ecs-init` 程序包。如果您的容器实例是从版本 `20190301` 或更高版本启动的，那么这些实例将包含所需版本的容器代理和 `ecs-init`。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

对于使用 Fargate 启动类型的任务，此参数需要任务或服务使用平台版本 1.3.0 或更高版本。

```
"dependsOn": [
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

类型：字符串

必需：是

必须符合指定条件的容器名称。

`condition`

类型：字符串

必需：是

容器的依赖项条件。以下是可用的条件及其行为：

- `START` – 此条件将立即模拟链接和卷的行为。它将验证从属容器是否是在允许其他容器启动前启动的。
- `COMPLETE` – 此条件将验证从属容器是否在允许其他容器启动前运行完成（退出）。这对于运行脚本然后退出的非主要容器非常有用。
- `SUCCESS` – 此条件与 `COMPLETE` 相同，但它还要求容器退出并具有 `zero` 状态。
- `HEALTHY` – 此条件将验证从属容器是否在允许其他容器启动前传递其 Docker 运行状况检查。这要求从属容器已配置运行状况检查。仅在任务启动时确认此条件。

容器超时

`startTimeout`

类型：整数

必需：否

在放弃解析容器的依赖项之前要等待的持续时间。例如，您在一个任务定义中指定两个容器，其中 containerA 需要依赖 containerB 达到 `COMPLETE`、`SUCCESS` 或 `HEALTHY` 状态。如果为 containerB 指定了一个 `startTimeout` 值并且它未在指定的超时时间内达到所需状态，则 containerA 将放弃并且不会启动。这会导致任务转换到 `STOPPED` 状态。

对于使用 EC2 启动类型的任务，容器实例需要至少 1.26.0 版的容器代理才能启用容器启动超时值。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。如果您使用 Amazon ECS-optimized Amazon Linux AMI，您的实例至少需要 1.26.0-1 版的 `ecs-init` 程序包。如果您的容器实例是从版本 `20190301` 或更高版本启动的，那么这些实例将包含所需版本的容器代理和 `ecs-init`。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

对于使用 Fargate 启动类型的任务，此参数需要任务或服务使用平台版本 1.3.0 或更高版本。

`stopTimeout`

类型：整数

必需：否

示例值：120

容器由于未自行正常退出而被强制终止前要等待的持续时间（以秒为单位）。对于使用 Fargate 启动类型的任务，最大停止超时值为 120 秒。对于使用 Fargate 启动类型的任务，此参数需要任务或服务使用平台版本 1.3.0 或更高版本。

对于使用 EC2 启动类型的任务，容器的停止超时值优先于 ECS_CONTAINER_STOP_TIMEOUT 容器代理配置参数（如果使用）。容器实例需要至少 1.26.0 版的容器代理才能启用容器停止超时值。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。如果您使用 Amazon ECS-optimized Amazon Linux AMI，您的实例至少需要 1.26.0-1 版的 ecs-init 程序包。如果您的容器实例是从版本 20190301 或更高版本启动的，那么这些实例将包含所需版本的容器代理和 ecs-init。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

系统控制

systemControls

类型：SystemControl 对象

必需：否

要在容器中设置的具有命名空间的内核参数的列表。此参数将映射到 Docker Remote API 的创建容器部分中的 Sysctls 以及 docker run 的 --sysctl 选项。

不建议在使用了 awsvpc 或 host 网络模式的单个任务中为多个容器指定与网络相关的 systemControls 参数，原因如下：

- 对于使用 awsvpc 网络模式的任务，如果为任何容器设置了 systemControls，则它将应用于该任务中的所有容器。如果您为单个任务中的多个容器设置了不同的 systemControls，最后启动的容器将确定哪个 systemControls 生效。
- 对于使用 host 网络模式的任务，不支持网络命名空间 systemControls。

如果您要设置 IPC 资源命名空间以用于任务中的容器，则以下内容将适用于您的系统控制。有关更多信息，请参阅 IPC 模式 (p. 79)。

- 对于使用 host IPC 模式的任务，不支持 IPC 命名空间 systemControls。
- 对于使用 task IPC 模式的任务，IPC 命名空间 systemControls 值将适用于任务中的所有容器。

  Note

  使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

```
"systemControls": [
    {
         "namespace":"string",
         "value":"string"
    }
]
```

namespace

类型：字符串

必需：否

需要为其设置 value 的具有命名空间的内核参数。

有效 IPC 命名空间值："kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"，以及以 "fs.mqueue.*" 开头的 Sysctls

有效网络命名空间值：以 "net.*" 开头的 Sysctls

value

类型：字符串

必需：否

在 namespace 中指定的具有命名空间的内核参数的值。

交互式

interactive

类型：布尔值

必需：否

当此参数为 true 时，您可以部署需要分配 stdin 或 tty 的容器化的应用程序。此参数将映射到 Docker Remote API 的创建容器部分中的 OpenStdin 以及 docker run 的 --interactive 选项。

伪终端

pseudoTerminal

类型：布尔值

必需：否

当此参数为 true 时，则分配 TTY。此参数将映射到 Docker Remote API 的创建容器部分中的 Tty 以及 docker run 的 --tty 选项。

卷

当您注册任务定义时，可以选择指定一个卷列表，这些卷将传递到容器实例上的 Docker 守护程序，然后它们将变得可供同一容器实例上的其他容器访问。

下面是可使用的数据卷的类型：

- Docker 卷 — 在容器实例上的 /var/lib/docker/volumes 下创建的 Docker 托管卷。Docker 卷驱动程序（也称为插件）用于将卷与外部存储系统（如 ）集成。可使用内置 local 卷驱动程序或第三方卷驱动程序。仅当使用 启动类型时，才支持 Docker 卷。Windows 容器仅支持使用 驱动程序。要使用 Docker 卷，请在任务定义中指定 。有关更多信息，请参阅使用卷。
- 绑定挂载 — 主机上的文件或目录将挂载到容器中。当使用 Fargate 或 启动类型时，支持绑定挂载主机卷。要使用绑定挂载主机卷，请在任务定义中指定 sourcePath 和可选的 值。有关更多信息，请参阅使用绑定挂载。

有关更多信息，请参阅 在任务中使用数据卷 (p. 83)。

容器定义中允许以下参数：

name

类型：字符串

必需：否

卷的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。此名称已在容器定义 `mountPoints` 的 `sourceVolume` 参数中引用。

`dockerVolumeConfiguration`

Type: Object

Required: No

This parameter is specified when using Docker volumes. Docker volumes are only supported when using the EC2 launch type. Windows containers only support the use of the `local` driver. To use bind mounts, specify a `host` instead.

`scope`

Type: String

Valid Values: `task` | `shared`

Required: No

The scope for the Docker volume, which determines its lifecycle. Docker volumes that are scoped to a `task` are automatically provisioned when the task starts and destroyed when the task stops. Docker volumes that are scoped as `shared` persist after the task stops.

`autoprovision`

Type: Boolean

Default value: `false`

Required: No

If this value is `true`, the Docker volume is created if it does not already exist. This field is only used if the `scope` is `shared`. If the `scope` is `task` then this parameter must either be omitted or set to `false`.

`driver`

Type: String

Required: No

The Docker volume driver to use. The driver value must match the driver name provided by Docker because it is used for task placement. If the driver was installed using the Docker plugin CLI, use `docker plugin ls` to retrieve the driver name from your container instance. If the driver was installed using another method, use Docker plugin discovery to retrieve the driver name. For more information, see Docker plugin discovery. This parameter maps to `Driver` in the Create a volume section of the Docker Remote API and the `--driver` option to `docker volume create`.

`driverOpts`

Type: String

Required: No

A map of Docker driver specific options to pass through. This parameter maps to `DriverOpts` in the Create a volume section of the Docker Remote API and the `--opt` option to `docker volume create`.

`labels`

Type: String

Required: No

Custom metadata to add to your Docker volume. This parameter maps to `Labels` in the Create a volume section of the Docker Remote API and the `--label` option to `docker volume create`.

`host`

Required: No

This parameter is specified when using bind mounts. To use Docker volumes, specify a `dockerVolumeConfiguration` instead. The contents of the `host` parameter determine whether your bind mount data volume persists on the host container instance and where it is stored. If the `host` parameter is empty, then the Docker daemon assigns a host path for your data volume, but the data is not guaranteed to persist after the containers associated with it stop running.

Bind mount host volumes are supported when using either the EC2 or Fargate launch types.

Windows containers can mount whole directories on the same drive as `$env:ProgramData`. Windows containers cannot mount directories on a different drive, and mount point cannot be across drives. For example, you can mount `C:\my\path:C:\my\path` and `D:\:D:\`, but not `D:\my\path:C:\my\path` or `D:\:C:\my\path`.

`sourcePath`

Type: String

Required: No

When the `host` parameter is used, specify a `sourcePath` to declare the path on the host container instance that is presented to the container. If this parameter is empty, then the Docker daemon has assigned a host path for you. If the `host` parameter contains a `sourcePath` file location, then the data volume persists at the specified location on the host container instance until you delete it manually. If the `sourcePath` value does not exist on the host container instance, the Docker daemon creates it. If the location does exist, the contents of the source path folder are exported.

任务放置约束

您在注册任务定义时，可以提供任务放置约束，自定义 Amazon ECS 如何放置任务。

如果您使用的是 Fargate 启动类型，则不支持任务放置约束。默认情况下，Fargate 任务分布在可用区中。

对于使用 EC2 启动类型的任务，则可以使用约束来根据可用区、实例类型或自定义属性来放置任务。有关更多信息，请参阅Amazon ECS 任务放置约束 (p. 257)。

容器定义中允许以下参数：

`expression`

类型：字符串

必需：否

应用于约束的集群查询语言表达式。有关更多信息，请参阅 集群查询语言 (p. 260)。

`type`

类型：字符串

必需：是

约束类型。使用 `memberOf` 将选择限制为一组有效的候选。

启动类型

当您注册任务定义时，您可以指定要用于任务的启动类型。有关更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。

任务定义中允许以下参数：

`requiresCompatibilities`

类型：字符串数组

必需：否

有效值：`EC2` | `FARGATE`

任务使用的启动类型。这将启用检查以确保任务定义中使用的所有参数均符合启动类型的要求。

有效值为 `FARGATE` 和 `EC2`。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。

任务大小

注册任务定义时，您可以指定用于任务的总 CPU 和内存量。这独立于容器定义级别的 `cpu` 和 `memory` 值。如果使用的是 EC2 启动类型，则这些字段是可选字段。如果使用的是 Fargate 启动类型，则这些字段是必填字段，并且 `cpu` 和 `memory` 具有受支持的特定值。

> **Note**
>
> Windows 容器将忽略任务级 CPU 和内存参数。我们建议为 Windows 容器指定容器级资源。

任务定义中允许以下参数：

`cpu`

类型：字符串

必需：否

> **Note**
>
> Windows 容器不支持此参数。

要为任务提供的 CPU 单位的硬限制。它在任务定义中可以表示为使用 CPU 单元时的整数（例如 `1024`），或表示为使用 vCPU 时的字符串（例如 `1 vCPU` 或 `1 vcpu`）。注册任务定义时，vCPU 值将转换为指示 CPU 单元的整数。

如果使用的是 EC2 启动类型，则此字段是可选字段。如果您的集群没有任何带可用的已请求 CPU 单位的已注册容器实例，则该任务将失败。受支持的值介于 `128` 个 CPU 单元 (`0.125` 个 vCPU) 和 `10240` 个 CPU 单元 (`10` 个 vCPU) 之间。

如果使用 Fargate 启动类型，此字段为必填字段，并且必须使用以下值之一，这决定了 `memory` 参数支持的值范围：

CPU 值	内存值 (MiB)
256 (.25 vCPU)	512 (0.5GB)、1024 (1GB)、2048 (2GB)
512 (.5 vCPU)	1024 (1GB)、2048 (2GB)、3072 (3GB)、4096 (4GB)

CPU 值	内存值 (MiB)
1024 (1 vCPU)	2048 (2GB)、3072 (3GB)、4096 (4GB)、5120 (5GB)、6144 (6GB)、7168 (7GB)、8192 (8GB)
2048 (2 vCPU)	4096 (4GB) 到 16384 (16GB) 之间（以 1024 (1GB) 为增量）
4096 (4 vCPU)	8192 (8GB) 到 30720 (30GB) 之间（以 1024 (1GB) 为增量）

`memory`

类型：字符串

必需：否

> **Note**
>
> Windows 容器不支持此参数。

要向任务提供的内存的硬限制（以 MiB 为单位）。它在任务定义中可以表示为使用 MiB 时的整数（例如 `1024`），或表示为使用 GB 时的字符串（例如 `1GB` 或 `1 GB`）。注册任务定义时，GB 值将转换为指示 MiB 的整数。

如果使用的是 EC2 启动类型，则此字段是可选字段。如果您的集群没有任何带可用的已请求内存的已注册容器实例，则该任务将失败。如果您尝试通过为任务提供尽可能多的用于特定实例类型的内存来最大程度地利用资源，请参阅容器实例内存管理 (p. 190)。

如果使用 Fargate 启动类型，此字段为必填字段，并且必须使用以下值之一，这决定了 `cpu` 参数支持的值范围：

内存值 (MiB)	CPU 值
512 (0.5GB)、1024 (1GB)、2048 (2GB)	256 (.25 vCPU)
1024 (1GB)、2048 (2GB)、3072 (3GB)、4096 (4GB)	512 (.5 vCPU)
2048 (2GB)、3072 (3GB)、4096 (4GB)、5120 (5GB)、6144 (6GB)、7168 (7GB)、8192 (8GB)	1024 (1 vCPU)
4096 (4GB) 到 16384 (16GB) 之间（以 1024 (1GB) 为增量）	2048 (2 vCPU)
8192 (8GB) 到 30720 (30GB) 之间（以 1024 (1GB) 为增量）	4096 (4 vCPU)

其他任务定义参数

以下任务定义参数可在 Amazon ECS 控制台中注册任务定义时使用，方法是使用 Configure via JSON (通过 JSON 配置) 选项。有关更多信息，请参阅创建任务定义 (p. 45)。

主题

- IPC 模式 (p. 79)
- PID 模式 (p. 79)

- 代理配置 (p. 79)

IPC 模式

`ipcMode`

类型：字符串

必需：否

用于任务中的容器的 IPC 资源命名空间。有效值为 `host`、`task` 或 `none`。如果指定了 `host`，则在同一容器实例上指定了 `host` IPC 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的 IPC 资源。如果指定了 `task`，则指定任务中的所有容器将共享相同的 IPC 资源。如果指定了 `none`，则任务的容器中的 IPC 资源是私有的，不与任务中或容器实例上的其他容器共享。如果未指定任何值，则 IPC 资源命名空间共享取决于容器实例上的 Docker 守护程序设置。有关更多信息，请参阅 Docker 运行参考中的 IPC 设置。

如果使用了 `host` IPC 模式，请注意，发生非预期的 IPC 命名空间公开的风险会提高。有关更多信息，请参阅 Docker 安全性。

如果您使用 `systemControls` 为任务中的容器设置具有命名空间的内核参数，则以下内容将适用于您的 IPC 资源命名空间。有关更多信息，请参阅系统控制 (p. 73)。

- 对于使用 `host` IPC 模式的任务，不支持与 IPC 命名空间相关的 `systemControls`。
- 对于使用 `task` IPC 模式的任务，与 IPC 命名空间相关的 `systemControls` 将适用于任务中的所有容器。

Note

使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

PID 模式

`pidMode`

类型：字符串

必需：否

用于任务中的容器的过程命名空间。有效值为 `host` 或 `task`。如果指定了 `host`，则在同一容器实例上指定了 `host` PID 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的 IPC 资源。如果指定了 `task`，则指定任务中的所有容器将共享相同的过程命名空间。如果未指定任何值，则默认值为私有命名空间。有关更多信息，请参阅 Docker 运行参考 中的 PID 设置。

如果使用了 `host` PID 模式，请注意，发生非预期的过程命名空间公开的风险会提高。有关更多信息，请参阅 Docker 安全性。

Note

使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

代理配置

`proxyConfiguration`

类型：ProxyConfiguration 对象

必需：否

App Mesh 代理的配置详细信息。

对于使用 EC2 启动类型的任务，容器实例需要至少 1.26.0 版的容器代理和至少 1.26.0-1 版的 `ecs-init` 程序包才能启用代理配置。如果您的容器实例是从经 Amazon ECS 优化的 AMI 版本 `20190301` 或更高版本启动的，那么这些实例将包含所需版本的容器代理和 `ecs-init`。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

对于使用 Fargate 启动类型的任务，此功能需要任务或服务使用平台版本 1.3.0 或更高版本。

> **Note**
>
> Windows 容器不支持此参数。

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
           "value": "string"
        }
    ]
}
```

`type`

类型：字符串

有效值：`APPMESH`

必需：否

代理类型。`APPMESH` 是唯一受支持的值。

`containerName`

类型：字符串

必需：是

将作为 App Mesh 代理的容器的名称。

`properties`

类型：KeyValuePair 对象的数组

必需：否

提供容器网络接口 (CNI) 插件的网络配置参数集，以键值对形式指定。

- `IgnoredUID` – （必填）代理容器的用户 ID (UID)，由容器定义中的 `user` 参数定义。此字段用于确保代理会忽略自己的流量。如果指定了 `IgnoredGID`，此字段可为空。
- `IgnoredGID` – （必填）代理容器的组 ID (GID)，由容器定义中的 `user` 参数定义。此字段用于确保代理会忽略自己的流量。如果指定了 `IgnoredUID`，此字段可为空。
- `AppPorts` – （必填）应用程序使用的端口的列表。发送到这些端口的网络流量将转发到 `ProxyIngressPort` 和 `ProxyEgressPort`。
- `ProxyIngressPort` – （必填）指定传入到 `AppPorts` 的流量将定向到的端口。
- `ProxyEgressPort` – （必填）指定从 `AppPorts` 传出的流量将定向到的端口。

- `EgressIgnoredPorts` – (必填) 进入这些指定端口的出口流量将被忽略但不会重定向到 `ProxyEgressPort`。它可以是空列表。
- `EgressIgnoredIPs` – (必填) 进入这些指定 IP 地址的出口流量将被忽略但不会重定向到 `ProxyEgressPort`。它可以是空列表。

`name`

类型：字符串

必需：否

键值对的名称。

`value`

类型：字符串

必需：否

键值对的值。

Amazon ECS 启动类型

Amazon ECS 启动类型决定了托管您的任务和服务的基础设施类型。

Fargate 启动类型

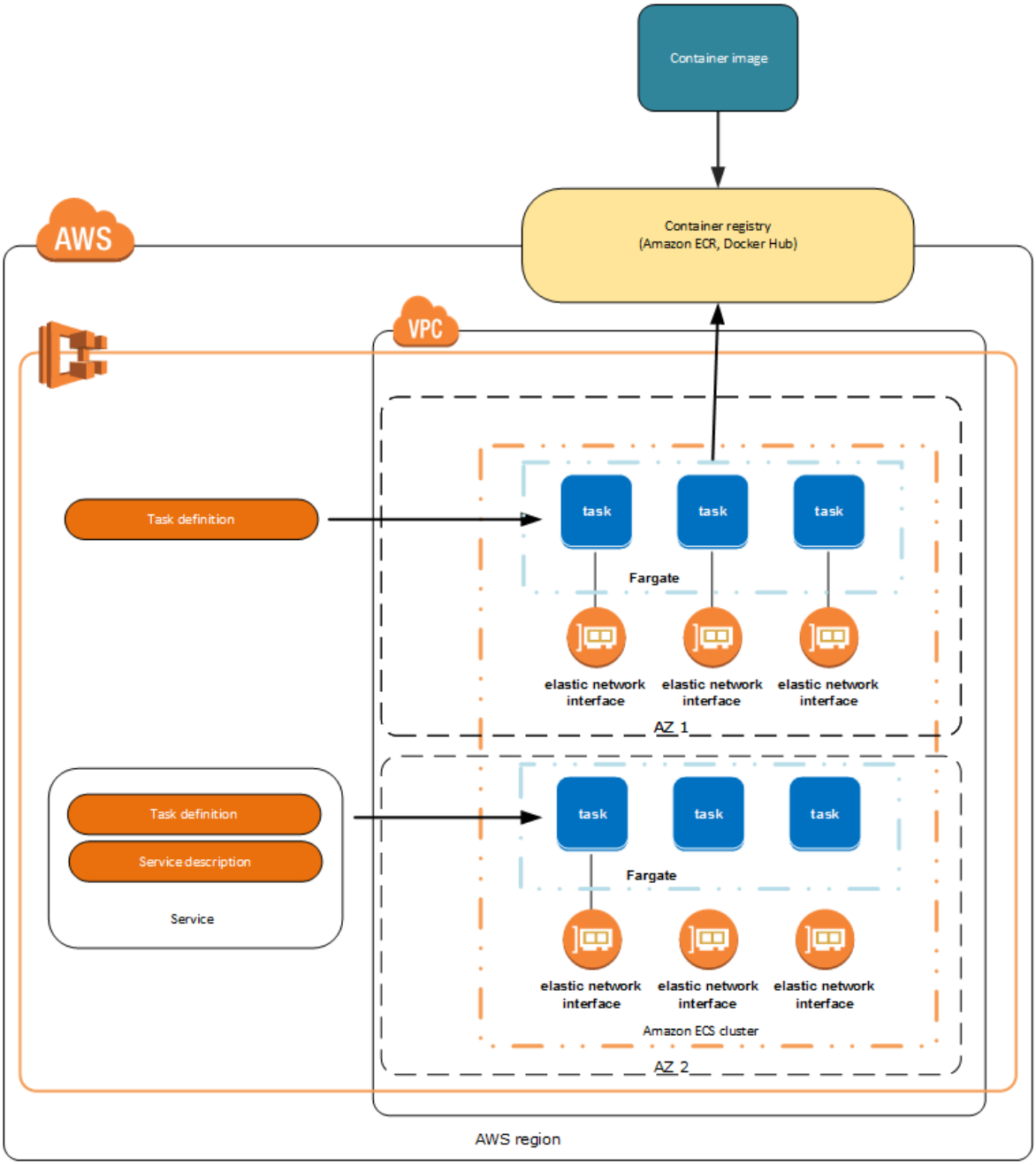
Fargate 启动类型允许您运行容器化的应用程序，而无需预置和管理后台基础设施。只需注册您的任务定义，Fargate 就会为您启动容器。

AWS Fargate 启动类型当前在以下区域可用：

区域名称	区域
美国东部 (弗吉尼亚北部)	us-east-1
美国东部 (俄亥俄)	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部 (俄勒冈)	us-west-2
亚太地区 (孟买)	ap-south-1
欧洲 (爱尔兰)	eu-west-1
欧洲 (伦敦)	eu-west-2
欧洲 (法兰克福)	eu-central-1
亚太地区 (东京)	ap-northeast-1
加拿大 (中部)	ca-central-1
亚太地区 (首尔)	ap-northeast-2
亚太地区 (新加坡)	ap-southeast-1

区域名称	区域
亚太地区 (悉尼)	ap-southeast-2

下图显示了一般架构：

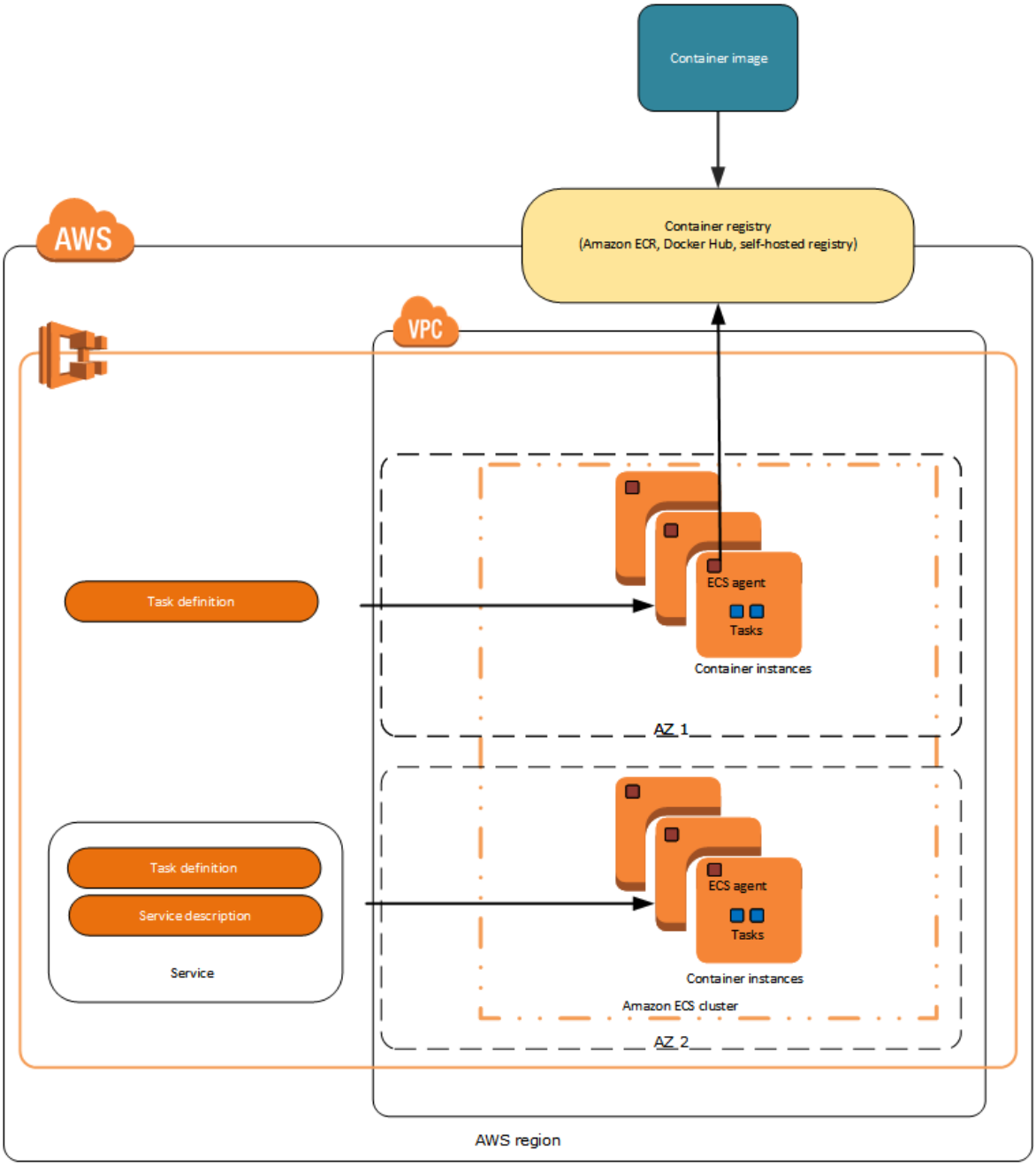


有关 Amazon ECS 以及 AWS Fargate 的更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。

EC2 启动类型

EC2 启动类型允许您在管理的 Amazon EC2 实例集群上运行容器化的应用程序。

下图显示了一般架构：



在任务中使用数据卷

有多个在 Amazon ECS 任务定义中使用数据卷的使用案例。我们提供以下指导，按启动类型分类。

Fargate 任务仅支持非持久性存储卷。有关更多信息，请参阅Fargate 任务存储 (p. 84)。

对于 EC2 任务，在以下常见示例中使用数据卷：

- 提供持久性数据卷以用于容器

- 定义空的非持久性数据卷并将其挂载到多个容器上
- 在同一个容器实例上不同容器中的不同位置共享定义的数据卷
- 向由第三方卷驱动程序管理的任务提供数据卷

卷的生命周期可以绑定到特定的任务或特定容器实例的生命周期。

下面是可使用的数据卷的类型：

- Docker 卷 — 在容器实例上的 `/var/lib/docker/volumes` 下创建的 Docker 托管卷。Docker 卷驱动程序（也称为插件）用于将卷与外部存储系统（如 ）集成。可使用内置 `local` 卷驱动程序或第三方卷驱动程序。仅当使用 启动类型时，才支持 Docker 卷。Windows 容器仅支持使用 驱动程序。要使用 Docker 卷，请在任务定义中指定 。有关更多信息，请参阅使用卷。
- 绑定挂载 — 主机上的文件或目录将挂载到容器中。当使用 Fargate 或 启动类型时，支持绑定挂载主机卷。要使用绑定挂载主机卷，请在任务定义中指定 `sourcePath` 和可选的 值。有关更多信息，请参阅使用绑定挂载。

> **Note**
>
> 在发布经 Amazon ECS 优化的 AMI 版本 2017.03.a 之前，只有在启动 Docker 守护程序时可用的文件系统才能用于 Docker 容器。您可以使用最新的经 Amazon ECS 优化的 AMI 以避免此限制，也可以将 `docker` 程序包升级到最新版本并重新启动 Docker。

主题

- Fargate 任务存储 (p. 84)
- Docker 卷 (p. 85)
- 绑定挂载 (p. 89)

Fargate 任务存储

在预配置时，每个 Fargate 任务都会收到以下存储。任务存储是短暂存储。Fargate 任务停止后，该存储将被删除。

- 10 GB 的 Docker 层存储
- 额外 4 GB 用于卷挂载。可在任务定义中使用 `volumes`、`mountPoints` 和 `volumesFrom` 参数挂载此存储和在容器之间共享此存储。

> **Note**
>
> `host` 和 `sourcePath` 参数不支持 Fargate 任务。

有关 Amazon ECS 默认服务限制的更多信息，请参阅 Amazon ECS 服务限制 (p. 540)。

为 Fargate 任务中的容器提供非持久性的空存储

在此示例中，您可能在任务期间具有需要访问同一临时文件存储位置的两个数据库容器。

1. 在任务定义 `volumes` 部分中，使用名称 `database_scratch` 定义卷。

```
"volumes": [ { "name": "database_scratch", "host": {} } ]
```

2. 在 `containerDefinitions` 部分中，创建数据库容器定义，以便它们挂载非持久性存储。

```
"containerDefinitions": [ { "name": "database1", "image": "my-repo/database",
"cpu": 100, "memory": 100, "essential": true, "mountPoints": [ { "sourceVolume":
```

```
 "database_scratch", "containerPath": "/var/scratch" } ] }, { "name": "database2",
 "image": "my-repo/database", "cpu": 100, "memory": 100, "essential": true,
 "mountPoints": [ { "sourceVolume": "database_scratch", "containerPath": "/var/
scratch" } ] } ]
```

Docker 卷

当使用 Docker 卷时，可以使用内置的 `local` 驱动程序或第三方卷驱动程序。Docker 卷由 Docker 管理，而目录在包含卷数据的容器实例上的 `/var/lib/docker/volumes` 中创建。

要使用 Docker 卷，请在任务定义中指定 `dockerVolumeConfiguration`。有关更多信息，请参阅使用卷。

Docker 卷的一些常见使用案例包括：

- 提供持久性数据卷以用于容器
- 在同一个容器实例上不同容器中的不同位置共享一个定义的数据卷
- 定义空的非持久性数据卷，并将其挂载到同一任务内的多个容器上
- 向由第三方驱动程序管理的任务提供数据卷

Docker 卷注意事项

使用 Docker 卷时应注意以下事项：

- 仅当使用 EC2 启动类型时，才支持 Docker 卷。
- Windows 容器仅支持使用 `local` 驱动程序。
- 如果使用第三方驱动程序，则应在容器代理启动之前在容器实例上安装并激活该驱动程序。如果在代理启动之前第三方驱动程序未处于活动状态，则可以使用下列命令之一重新启动容器代理：
  - 对于Amazon ECS-optimized Amazon Linux 2 AMI：

    ```
    sudo systemctl restart ecs
    ```

  - 对于Amazon ECS-optimized Amazon Linux AMI：

    ```
    sudo stop ecs && sudo start ecs
    ```

在任务定义中指定 Docker 卷

在您的容器可以使用数据卷之前，您必须在任务定义中指定卷和挂载点配置。此部分描述容器的卷配置。对于使用 Docker 卷的任务，请指定 `dockerVolumeConfiguration`。对于使用绑定挂载主机卷的任务，请指定 `host` 和可选的 `sourcePath`。

下面显示的任务定义 JSON 显示容器的 `volumes` 和 `mountPoints` 对象的语法。

```
{
    "containerDefinitions": [
        {
            "mountPoints": [
                {
                    "sourceVolume": "string",
                    "containerPath": "/path/to/mount_volume",
                    "readOnly": boolean
                }
```

```
            ]
        }
    ],
    "volumes": [
        {
            "name": "string",
            "dockerVolumeConfiguration": {
                "scope": "string",
                "autoprovision": boolean,
                "driver": "string",
                "driverOpts": {
                    "key": "value"
                },
                "labels": {
                    "key": "value"
                }
            }
        }
    ]
}
```

`name`

类型：字符串

必需：否

卷的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。此名称已在容器定义 `mountPoints` 的 `sourceVolume` 参数中引用。

`dockerVolumeConfiguration`

Type: Object

Required: No

This parameter is specified when using Docker volumes. Docker volumes are only supported when using the EC2 launch type. Windows containers only support the use of the `local` driver. To use bind mounts, specify a `host` instead.

`scope`

Type: String

Valid Values: `task` | `shared`

Required: No

The scope for the Docker volume, which determines its lifecycle. Docker volumes that are scoped to a `task` are automatically provisioned when the task starts and destroyed when the task stops. Docker volumes that are scoped as `shared` persist after the task stops.

`autoprovision`

Type: Boolean

Default value: `false`

Required: No

If this value is `true`, the Docker volume is created if it does not already exist. This field is only used if the `scope` is `shared`. If the `scope` is `task` then this parameter must either be omitted or set to `false`.

`driver`

Type: String

Required: No

The Docker volume driver to use. The driver value must match the driver name provided by Docker because it is used for task placement. If the driver was installed using the Docker plugin CLI, use `docker plugin ls` to retrieve the driver name from your container instance. If the driver was installed using another method, use Docker plugin discovery to retrieve the driver name. For more information, see Docker plugin discovery. This parameter maps to `Driver` in the Create a volume section of the Docker Remote API and the `--driver` option to `docker volume create`.

`driverOpts`

Type: String

Required: No

A map of Docker driver specific options to pass through. This parameter maps to `DriverOpts` in the Create a volume section of the Docker Remote API and the `--opt` option to `docker volume create`.

`labels`

Type: String

Required: No

Custom metadata to add to your Docker volume. This parameter maps to `Labels` in the Create a volume section of the Docker Remote API and the `--label` option to `docker volume create`.

`mountPoints`

Type: Object

Required: No

The mount points for data volumes in your container.

This parameter maps to `Volumes` in the Create a container section of the Docker Remote API and the `--volume` option to docker run.

Windows containers can mount whole directories on the same drive as `$env:ProgramData`. Windows containers cannot mount directories on a different drive, and mount point cannot be across drives.

`sourceVolume`

Type: String

Required: Yes, when `mountPoints` are used

The name of the volume to mount.

`containerPath`

Type: String

Required: Yes, when `mountPoints` are used

The path on the container to mount the volume at.

`readOnly`

Type: Boolean

Required: No

If this value is `true`, the container has read-only access to the volume. If this value is `false`, then the container can write to the volume. The default value is `false`.

示例

以下示例显示 Docker 卷的使用。

使用 Docker 卷为容器提供非持久性存储

在此示例中，您希望容器使用一个空数据卷，但您不希望在完成任务后保留该卷。例如，您可能具有一个需要在任务期间访问某个临时文件存储位置的容器。可以使用 Docker 卷实现此任务。

1. 在任务定义 `volumes` 部分中，使用 `name` 和 `DockerVolumeConfiguration` 值定义数据卷。在此示例中，我们将范围指定为 `task`，以便在任务停止后删除此卷并使用内置的 `local` 驱动程序。

```
"volumes": [
    {
        "name": "scratch",
        "dockerVolumeConfiguration" : {
            "scope": "task",
            "driver": "local",
            "labels": {
                "scratch": "space"
            }
        }
    }
]
```

2. 在 `containerDefinitions` 部分中，使用引用已定义卷的名称的 `mountPoints` 值和将卷挂载到容器上所在的 `containerPath` 值定义容器。

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
            {
              "sourceVolume": "scratch",
              "containerPath": "/var/scratch"
            }
        ]
    }
]
```

使用 Docker 卷为容器提供持久性存储

在本示例中，您希望多个容器使用一个共享卷，并希望在使用此卷的任何单个任务停止后该卷持续存在。正在使用的是内置的 `local` 驱动程序，以便该卷仍绑定到容器实例的生命周期。

1. 在任务定义 `volumes` 部分中，使用 `name` 和 `DockerVolumeConfiguration` 值定义数据卷。在此示例中，指定 `shared` 范围以便此卷持久存在，将自动预置设置为 `true` 以便创建此卷进行使用，并使用内置的 `local` 驱动程序。

```
"volumes": [
    {
        "name": "database",
        "dockerVolumeConfiguration" : {
            "scope": "shared",
```

```
            "autoprovision": true,
            "driver": "local",
            "labels": {
                "database": "database_name"
            }
        }
    }
]
```

2. 在 containerDefinitions 部分中，使用引用已定义卷的名称的 mountPoints 值和将卷挂载到容器上所在的 containerPath 值定义容器。

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    },
    {
      "name": "container-2",
      "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    }
  ]
```

绑定挂载

使用绑定装载，主机上的文件或目录将装载到容器中。使用 EC2 或 Fargate 启动类型时，支持绑定挂载主机卷。Fargate 任务仅支持非持久存储卷，因此不支持 host 和 sourcePath 字段。有关更多信息，请参阅Fargate 任务存储 (p. 84)。

要将绑定挂载主机卷与使用 EC2 启动类型的任务结合使用，请在任务定义中指定 host 和可选的 sourcePath 值。有关详细信息，请参阅使用绑定挂载。

绑定挂载的一些常见使用案例包括：

- 提供持久性数据卷以用于容器
- 定义空的非持久性数据卷，并将其挂载到同一容器实例上的多个容器
- 在同一个容器实例上不同容器中的不同位置共享定义的数据卷

在任务定义中指定绑定挂载

在您的容器可以使用绑定挂载主机卷之前，您必须在任务定义中指定卷和挂载点配置。此部分描述容器的卷配置。对于使用绑定挂载主机卷的任务，请指定 host 值和可选的 sourcePath 值。

以下任务定义 JSON 代码段显示容器的 volumes 和 mountPoints 对象的语法：

```
{
   "family": "",
   ...
```

```
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
               "containerPath" : "/path/to/mount_volume",
               "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
          "host" : {
             "sourcePath" : "string"
          },
          "name" : "string"
       }
    ]
}
```

`name`

类型：字符串

必需：否

卷的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。此名称已在容器定义 `mountPoints` 的 `sourceVolume` 参数中引用。

`host`

Required: No

This parameter is specified when using bind mounts. To use Docker volumes, specify a `dockerVolumeConfiguration` instead. The contents of the `host` parameter determine whether your bind mount data volume persists on the host container instance and where it is stored. If the `host` parameter is empty, then the Docker daemon assigns a host path for your data volume, but the data is not guaranteed to persist after the containers associated with it stop running.

Bind mount host volumes are supported when using either the EC2 or Fargate launch types.

Windows containers can mount whole directories on the same drive as `$env:ProgramData`. Windows containers cannot mount directories on a different drive, and mount point cannot be across drives. For example, you can mount `C:\my\path:C:\my\path` and `D:\:D:\`, but not `D:\my\path:C:\my\path` or `D:\:C:\my\path`.

`sourcePath`

Type: String

Required: No

When the `host` parameter is used, specify a `sourcePath` to declare the path on the host container instance that is presented to the container. If this parameter is empty, then the Docker daemon has assigned a host path for you. If the `host` parameter contains a `sourcePath` file location, then the data volume persists at the specified location on the host container instance until you delete it manually. If the `sourcePath` value does not exist on the host container instance, the Docker daemon creates it. If the location does exist, the contents of the source path folder are exported.

`mountPoints`

Type: Object

Required: No

The mount points for data volumes in your container.

This parameter maps to `Volumes` in the Create a container section of the Docker Remote API and the `--volume` option to docker run.

Windows containers can mount whole directories on the same drive as `$env:ProgramData`. Windows containers cannot mount directories on a different drive, and mount point cannot be across drives.

`sourceVolume`

Type: String

Required: Yes, when `mountPoints` are used

The name of the volume to mount.

`containerPath`

Type: String

Required: Yes, when `mountPoints` are used

The path on the container to mount the volume at.

`readOnly`

Type: Boolean

Required: No

If this value is `true`, the container has read-only access to the volume. If this value is `false`, then the container can write to the volume. The default value is `false`.

示例

使用绑定挂载为容器提供非持久性的空存储

在某些情况下，您希望容器共享同一空数据卷，但您不希望在完成任务后保留该数据。例如，您可能在任务期间具有需要访问同一临时文件存储位置的两个数据库容器。此任务可以使用 Docker 卷或绑定挂载主机卷来实现。

1. 在任务定义 `volumes` 部分中，使用名称 `database_scratch` 定义绑定挂载。

   Note

   由于 `database_scratch` 绑定挂载未指定源路径，因此 Docker 守护程序会为您管理此绑定挂载。如果任何容器均未引用此绑定挂载，则 Amazon ECS 容器代理任务清除服务最终会将其删除（默认情况下，此情况会在容器存在 3 小时后出现，但您可以使用 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 代理变量配置此持续时间）。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。如果您需要保留此数据，请为绑定挂载指定 `sourcePath` 值。

```
"volumes": [
  {
    "name": "database_scratch",
    "host": {}
  }
]
```

2. 在 `containerDefinitions` 部分中，创建数据库容器定义，以便它们挂载非持久性存储。

```
"containerDefinitions": [
  {
    "name": "database1",
    "image": "my-repo/database",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "database_scratch",
        "containerPath": "/var/scratch"
      }
    ]
  },
  {
    "name": "database2",
    "image": "my-repo/database",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "database_scratch",
        "containerPath": "/var/scratch"
      }
    ]
  }
]
```

使用绑定挂载为容器提供持久性存储

当使用绑定挂载时，如果指定 `sourcePath` 值，则数据即使在引用它的所有容器停止后也将保留。存在于 `sourcePath` 的任何文件将提供给位于 `containerPath` 值的容器，而且写入 `containerPath` 值的任何文件将写入容器实例上的 `sourcePath` 值。

> **Important**
>
> Amazon ECS 不会跨容器实例同步存储。使用持久性存储的任务可置于您的集群中具有可用容量的任何容器实例上。如果任务在停止和重新启动后需要持久性存储，则您应始终在使用 AWS CLI start-task 命令启动任务时指定相同的容器实例。

1. 在任务定义 `volumes` 部分中，使用 `name` 和 `sourcePath` 值定义绑定挂载。

```
"volumes": [
  {
    "name": "webdata",
    "host": {
      "sourcePath": "/ecs/webdata"
    }
  }
]
```

2. 在 `containerDefinitions` 部分中，使用引用已定义绑定挂载的名称的 `mountPoints` 值和要在容器上挂载绑定挂载的 `containerPath` 值定义容器。

```
"containerDefinitions": [
  {
    "name": "web",
    "image": "nginx",
    "cpu": 99,
    "memory": 100,
```

```
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webdata",
          "containerPath": "/usr/share/nginx/html"
        }
      ]
    }
  ]
```

在多个容器上挂载定义的卷

您可以在任务定义中定义数据卷并在其他容器上的不同位置挂载该卷。例如，您的主机容器具有位于 `/data/webroot` 的网站数据文件夹，并且您可能想在具有不同文档根目录的两个不同 Web 服务器上以只读方式挂载该数据卷。

1. 在任务定义 `volumes` 部分中，使用名称 `webroot` 和源路径 `/data/webroot` 定义数据卷。

```
"volumes": [
  {
    "name": "webroot",
    "host": {
      "sourcePath": "/data/webroot"
    }
  }
]
```

2. 在 `containerDefinitions` 部分中，使用将 `mountPoints` 卷与指向容器的文档根目录的 `webroot` 值关联的 `containerPath` 值为每个 Web 服务器定义容器。

```
"containerDefinitions": [
  {
    "name": "web-server-1",
    "image": "my-repo/ubuntu-apache",
    "cpu": 100,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webroot",
        "containerPath": "/var/www/html",
        "readOnly": true
      }
    ]
  },
  {
    "name": "web-server-2",
    "image": "my-repo/sles11-apache",
    "cpu": 100,
    "memory": 100,
```

```
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/srv/www/htdocs",
          "readOnly": true
        }
      ]
    }
  ]
```

使用 volumesFrom 从其他容器挂载卷

您可以在一个容器上定义一个或多个卷，然后在不同的容器定义中使用 volumesFrom 参数（相同任务内）从其初始定义的挂载点的 sourceContainer 中挂载所有卷。volumesFrom 参数适用于任务定义中定义的卷以及使用 Dockerfile 内置到映像中的卷。

1. （可选）要共享内置到映像中的卷，您需要使用在 VOLUME 指令中声明的卷构建映像。以下示例 Dockerfile 使用了 httpd 映像，然后添加卷并在 Apache 文档根目录 (由 httpd Web 服务器使用的文件夹) 中的 dockerfile_volume 处挂载卷：

   ```
   FROM httpd
   VOLUME ["/usr/local/apache2/htdocs/dockerfile_volume"]
   ```

   您可以使用此 Dockerfile 生成映像并将其推送至存储库 (如 Docker Hub)，然后在任务定义中使用它。以下步骤中使用的示例 my-repo/httpd_dockerfile_volume 映像是通过上述 Dockerfile 构建的。
2. 创建一个为您的容器定义其他卷和挂载点的任务定义。在此示例 volumes 部分中，您将创建一个名为 empty 的空卷，此卷由 Docker 守护程序管理。此外，还将定义一个称为 host_etc 的主机卷，该卷导出主机容器实例上的 /etc 文件夹。

   ```
   {
     "family": "test-volumes-from",
     "volumes": [
       {
         "name": "empty",
         "host": {}
       },
       {
         "name": "host_etc",
         "host": {
           "sourcePath": "/etc"
         }
       }
     ],
   ```

   在容器定义部分中，创建一个挂载之前定义的卷的容器。在此示例中，web 容器（此容器使用通过 Dockerfile 中的卷构建的映像）挂载 empty 和 host_etc 卷。

   ```
   "containerDefinitions": [
       {
         "name": "web",
         "image": "my-repo/httpd_dockerfile_volume",
         "cpu": 100,
   ```

```
      "memory": 500,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "empty",
          "containerPath": "/usr/local/apache2/htdocs/empty_volume"
        },
        {
          "sourceVolume": "host_etc",
          "containerPath": "/usr/local/apache2/htdocs/host_etc"
        }
      ],
      "essential": true
    },
```

创建另一个容器，该容器使用 `volumesFrom` 挂载与 `web` 容器关联的所有卷。`web` 容器上的所有卷也同样会挂载到 `busybox` 容器上（包括 Dockerfile 中指定的用于构建 `my-repo/httpd_dockerfile_volume` 映像的卷）。

```
    {
      "name": "busybox",
      "image": "busybox",
      "volumesFrom": [
        {
          "sourceContainer": "web"
        }
      ],
      "cpu": 100,
      "memory": 500,
      "entryPoint": [
        "sh",
        "-c"
      ],
      "command": [
        "echo $(date) > /usr/local/apache2/htdocs/empty_volume/date && echo $(date)
 > /usr/local/apache2/htdocs/host_etc/date && echo $(date) > /usr/local/apache2/htdocs/
dockerfile_volume/date"
      ],
      "essential": false
    }
  ]
}
```

当此任务运行时，这两个容器将挂载卷，并且 `busybox` 容器中的 `command` 会将日期和时间写入每个卷文件夹中称为 `date` 的文件。这些文件夹随后将显示在由 `web` 容器显示的网站上。

> **Note**
>
> 由于 `busybox` 容器运行快速命令然后退出，它必须在容器定义中设置为 `"essential": false`。否则，它在退出时会停止整个任务。

awsvpc 网络模式的任务联网

`awsvpc` 网络模式提供的任务联网功能使 Amazon ECS 任务具有与 Amazon EC2 实例相同的联网属性。当您在任务定义中使用 `awsvpc` 网络模式时，每个从该任务定义启动的任务都会获取其自己的弹性网络接口

(ENI)、主要私有 IP 地址和内部 DNS 主机名。任务联网功能简化了容器联网，使您可以更好地控制容器化应用程序如何与您的 VPC 内的其他服务进行通信。

> **Note**
>
> 有关对任务可用的其他网络模式的信息，请参阅 网络模式 (p. 52)。

通过允许您在任务中更精细地使用安全组和网络监视工具，任务联网还为您的容器提供了更高的安全性。由于每个任务都有自己的 ENI，因此您也可以利用其他 Amazon EC2 联网功能（例如 VPC 流日志），以便您可以监控您任务的流量。此外，属于同一任务的容器可以通过 `localhost` 接口进行通信。一个任务在给定的时间只能有一个与之关联的 ENI。

创建的任务 ENI 由 Amazon ECS 完全托管。Amazon ECS 创建 ENI 并将它附加到具有指定安全组的容器实例。任务在 ENI 上发送和接收网络流量的方式与 Amazon EC2 实例使用其主要网络接口的方式相同。这些 ENI 在您账户的 Amazon EC2 控制台中可见，但它们不能由您的账户手动分离或修改。这是为了防止意外删除与正在运行的任务关联的 ENI。您可以在 Amazon ECS 控制台中或使用 DescribeTasks API 操作查看任务的 ENI 连接信息。当任务停止或服务缩减时，将分离并删除任务 ENI。

如果您的账户、IAM 用户或角色已选择使用 `awsvpcTrunking` 账户设置，并且您已启动具有增加的 ENI 密度的容器实例，则 Amazon ECS 还会为您的容器实例创建和附加“中继”网络接口。中继网络接口完全由 Amazon ECS 管理。在您从 Amazon ECS 集群中终止或取消注册容器实例时，将删除中继 ENI。有关选择使用 `awsvpcTrunking` 账户设置的更多信息，请参阅 使用具有增加的 ENI 限制的容器实例 (p. 176)。

任务联网方面的考量

使用任务联网时，需要考虑多个因素。

- 使用 `awsvpc` 网络模式的任务和服务需要与 Amazon ECS 服务相关的角色，从而向 Amazon ECS 提供代表您调用其他 AWS 服务的权限。此角色是在创建集群时 (或者在 AWS 管理控制台 中创建或更新服务时) 自动为您创建的。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色 (p. 392)。您也可以使用以下 AWS CLI 命令创建服务相关的角色：

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- Amazon ECS 允许使用具有增加的 ENI 密度的受支持的 Amazon EC2 实例类型来启动容器实例。当您使用这些实例类型并选择使用 `awsvpcTrunking` 账户设置时，新启动的容器实例将具有更高的 ENI 限制。此配置允许您在每个容器实例上放置更多任务。有关选择使用 `awsvpcTrunking` 账户设置的更多信息，请参阅 账户设置 (p. 120)。有关 ENI 中继的更多信息，请参阅 弹性网络接口中继 (p. 175)。
- 您的 Amazon ECS 容器实例需要至少 1.15.0 版的容器代理才能启用任务联网。要利用增加的 ENI 密度与中继功能，容器实例至少需要 1.28.1 版本的容器代理。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。如果您使用的是 Amazon ECS-optimized AMI，则实例至少需要 1.15.0-4（或面向 ENI 中继功能的 1.28.1-2）版本的 `ecs-init` 程序包。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。
- 目前，只有Amazon ECS-optimized AMI 的 Linux 变体，或具有 `ecs-init` 程序包的其他 Amazon Linux 变体支持任务联网。
- 对于使用 EC2 启动类型的任务，`awsvpc` 网络模式不提供具有公有 IP 地址的任务 ENI。要访问 Internet，必须在配置为使用 NAT 网关的私有子网中启动使用 EC2 启动类型的任务。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关 。入站网络访问必须使用私有 IP 地址或 DNS 主机名从 VPC 内进行，或者通过 VPC 内的负载均衡器进行路由。公有子网内启动的任务不具有出站网络访问权限。

  > **Note**
  >
  > 上述限制不适用于使用 Fargate 启动类型的任务。您可以对这些任务进行配置，以接收公有 IP 地址。

- 每个使用 `awsvpc` 网络模式的 Amazon ECS 任务都会接收到自己的 ENI，该 ENI 将附加到托管它的容器实例。可以附加到 Amazon EC2 实例的网络接口的数量有默认限制，并且主网络接口计为一个。例如，默认情况下，最多可将三个 ENI 附加到一个 `c5.large` 实例。实例的主网络接口计为一个，因此您可以向该

实例再附加 2 个 ENI。由于每个使用 awsvpc 网络模式的任务均需一个 ENI，因此，您通常只能在该实例类型上运行 2 个此类任务。有关每个实例类型的默认 ENI 限制的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的每个实例类型的每个网络接口的 IP 地址。

- Amazon ECS 支持使用受支持的 Amazon EC2 实例类型启动已增加 ENI 密度的容器实例。当您使用这些实例类型并选择使用 awsvpcTrunking 账户设置时，其他 ENI 将在新启动的容器实例上可用。此配置允许您使用 awsvpc 网络模式在每个容器实例上放置更多任务。通过使用此功能，启用了 awsvpcTrunking 的 c5.large 实例的 ENI 限制提高了 12。容器实例将具有主网络接口，而 Amazon ECS 将创建一个“中继”网络接口并将此接口附加到容器实例。因此，此配置允许您在容器实例上启动 10 个任务，而不是当前的两个任务。有关更多信息，请参阅弹性网络接口中继 (p. 175)。
- 在运行任务或创建使用 awsvpc 网络模式的服务时，限制为可在 awsvpcConfiguration 中指定 16 个子网和 5 个安全组。有关更多信息，请参阅 Amazon Elastic Container Service API Reference 中的 AwsVpcConfiguration。
- Amazon ECS 仅考虑附加到您的容器实例的 ENI。如果已手动将 ENI 附加到容器实例，则 Amazon ECS 可能会尝试在可用网络适配器附件太少的情况下将任务放置在实例上。在这种情况下，任务会超时，将从 PROVISIONING 变为 DEPROVISIONING，然后变为 STOPPED。我们建议您不要手动将 ENI 附加到容器实例。
- 容器实例必须使用 ecs.capability.task-eni 注册，才能考虑使用 awsvpc 网络模式放置任务。运行版本 1.15.0-4 或更高版本的 ecs-init 的容器实例注册有此属性。
- 创建并附加到容器实例的 ENI 不能由您的账户手动分离或修改。这是为了防止意外删除与正在运行的任务关联的 ENI。要释放任务的 ENI，请停止该任务。
- 在用 awsvpc 网络模式启动任务时，Amazon ECS 容器代理会为每个任务创建一个附加的 pause 容器，然后在任务定义中启动该容器。然后，它通过执行 amazon-ecs-cni-plugins CNI 插件来配置 pause 容器的网络命名空间。然后，该代理会启动任务中的其余容器，以使其共享 pause 容器的网络堆栈。这意味着，一个任务中的所有容器均可通过 ENI 的 IP 地址来寻址，并且这些容器可以通过 localhost 接口相互通信。
- 具有使用 awsvpc 网络模式的任务（例如，具有 Fargate 启动类型的任务）的服务仅支持 应用程序负载均衡器 和 网络负载均衡器；不支持 传统负载均衡器。此外，当您为这些服务创建任何目标组时，必须选择 ip 而不是 instance 作为目标类型。这是因为使用 awsvpc 网络模式的任务与 ENI 而不是 Amazon EC2 实例关联。有关更多信息，请参阅服务负载均衡 (p. 284)。
- 如果 VPC 已更新，例如更改所使用的 DHCP 选项集，并且您希望在使用该 VPC 的任务中使更改生效，则必须停止这些任务并启动新任务。

启用任务联网

要让任务使用任务联网，您必须在任务定义中指定 awsvpc 网络模式。有关更多信息，请参阅网络模式 (p. 52)。然后，当您运行任务或创建服务时，请指定网络配置，其中包含要放置任务的一个或多个子网以及要附加到其关联的 ENI 的一个或多个安全组。这些任务放置在与这些子网位于相同可用区中的有效容器实例上，并且指定的安全组与为该任务预配置的 ENI 关联。

使用 awslogs 日志驱动程序

您可在任务中配置容器以将日志信息发送到 CloudWatch Logs。如果您对任务使用 Fargate 启动类型，这允许您从容器查看日志。如果使用 EC2 启动类型，这使您能够在一个方便的位置查看容器中的不同日志，并防止您的容器日志占用您容器实例上的磁盘空间。本主题将帮助您在任务定义中开始使用 awslogs 日志驱动程序。

> **Note**
>
> 任务的容器所记录的信息类型主要取决于其 ENTRYPOINT 命令。默认情况下，捕获的日志显示命令输出是您在本地运行容器时在交互式终端上通常看到的内容，即 STDOUT 和 STDERR I/O 流。awslogs 日志驱动程序只是将 Docker 中的这些日志传递到 CloudWatch。有关如何处理

Docker 日志的更多信息，包括捕获不同文件数据或流的替代方法，请参阅 Docker 文档中的查看容器或服务的日志。

要将系统日志从 Amazon ECS 容器实例发送到 CloudWatch Logs，请参阅将 CloudWatch Logs 用于容器实例 (p. 181)。有关 CloudWatch Logs 的更多信息，请参阅 Amazon CloudWatch 用户指南 中的监控日志文件。

主题

- 在容器上启用 awslogs 日志驱动程序 (p. 98)
- 创建您的日志组 (p. 98)
- 可用的 awslogs 日志驱动程序选项 (p. 99)
- 在任务定义中指定日志配置 (p. 101)
- 在 CloudWatch Logs 中查看 awslogs 容器日志 (p. 102)

在容器上启用 awslogs 日志驱动程序

如果您对任务使用 Fargate 启动类型，要启用 `awslogs` 日志驱动程序，只需要将所需的 `logConfiguration` 参数添加到您的任务定义即可。此日志驱动程序会将您的 Fargate 任务配置为向 Amazon CloudWatch Logs 发送日志信息。有关更多信息，请参阅 在任务定义中指定日志配置 (p. 101)。

如果对任务使用 EC2 启动类型，要启用 `awslogs` 日志驱动程序，Amazon ECS 容器实例至少需要容器代理的 1.9.0 版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

Note

如果对容器实例使用的不是针对 Amazon ECS 优化过的 AMI（至少具有 1.9.0-1 版本的 `ecs-init` 程序包），则还需要指定当您通过在 docker run 语句或环境变量文件中使用以下环境变量启动代理时，`awslogs` 日志记录驱动程序在容器实例上可用。有关更多信息，请参阅安装 Amazon ECS 容器代理 (p. 198)。

```
ECS_AVAILABLE_LOGGING_DRIVERS='["json-file","awslogs"]'
```

Amazon ECS 容器实例还需要针对您用于启动容器实例的 IAM 角色的 `logs:CreateLogStream` 和 `logs:PutLogEvents` 权限。如果在 Amazon ECS 中启用 `awslogs` 日志驱动程序支持之前创建了 Amazon ECS 容器实例，则可能需要添加此权限。如果您的容器实例使用容器实例的托管 IAM 策略，则您的容器实例应具有正确的权限。有关检查 Amazon ECS 容器实例角色并为容器实例附加托管 IAM 策略的信息，请参阅在 IAM 控制台中检查 `ecsInstanceRole` (p. 405)。

创建您的日志组

`awslogs` 日志驱动程序可将日志流发送到 CloudWatch Logs 中的现有日志组，但无法创建日志组。您应确保创建您希望您的容器使用的日志组，然后您才能启动使用 `awslogs` 日志驱动程序的任何任务。该控制台提供了一个自动配置选项。如果您在控制台中注册任务定义并选择 Auto-configure CloudWatch Logs (自动配置 CloudWatch Logs) 选项，系统将为您创建日志组。或者，您可以使用以下步骤手动创建日志组。

例如，您可具有包含链接到一个 MySQL 容器（使用 `awslogs-mysql` 日志组）的 WordPress 容器（使用 `awslogs-wordpress` 日志组）的任务。下面几节说明如何使用 AWS CLI 和 CloudWatch 控制台创建这些日志组。

使用 AWS CLI 创建日志组

AWS Command Line Interface (AWS CLI) 是用于管理 AWS 服务的统一工具。只通过一个工具进行下载和配置，您就可以使用命令行控制多个 AWS 服务并利用脚本来自动执行这些服务。有关更多信息，请参见 AWS Command Line Interface 用户指南。

如果您安装了可以正常运行的 AWS CLI，则可使用它创建日志组。下面的命令将在 `us-west-2` 区域中创建一个名为 `awslogs-wordpress` 的日志组。为要创建的每个日志组运行此命令，将日志组名称替换为您的值和所需日志目标的区域名称。

```
aws logs create-log-group --log-group-name awslogs-wordpress --region us-west-2
```

使用自动配置功能创建日志组

在 Amazon ECS 控制台中注册任务定义时，您可以选择允许 Amazon ECS 自动配置 CloudWatch 日志。此选项还为您创建指定的日志组。为使此过程变得轻松，自动配置选项使用指定的前缀设置 CloudWatch 日志和日志组。

在 Amazon ECS 控制台中创建日志组

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在左侧导航窗格中，依次选择 Task Definitions、Create new Task Definition。
3. 选择兼容性选项，然后选择下一步。
4. 选择 Add container。
5. 在 Storage and Logging (存储和日志记录) 部分，对于 Log configuration (日志配置)，选择 Auto-configure CloudWatch Logs (自动配置 CloudWatch 日志)。
6. 输入您的 awslogs 日志驱动程序选项。有关更多信息，请参阅 在任务定义中指定日志配置 (p. 101)。
7. 完成任务定义向导的其余部分。

使用 CloudWatch 控制台创建日志组

以下步骤将在 CloudWatch 控制台中创建日志组。

在 CloudWatch 控制台中创建日志组

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，选择 Logs。
3. 选择 Actions、Create log group。
4. 对于 Log Group Name，输入要创建的日志组的名称。
5. 选择 Create log group 以完成操作。

可用的 awslogs 日志驱动程序选项

`awslogs` 日志驱动程序支持 Amazon ECS 任务定义中的下列选项。有关更多信息，请参阅 CloudWatch Logs 日志记录驱动程序。

`awslogs-create-group`

必需：否

指定您是否希望自动创建日志组。如果未指定此选项，则默认为 `false`。

> **Note**
>
> IAM 策略必须包含 `logs:CreateLogGroup` 权限，然后才能尝试使用 `awslogs-create-group`。

`awslogs-datetime-format`

必需：否

此选项以 Python `strftime` 格式定义多行开始位置模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。因此，匹配行是日志消息之间的分隔符。

使用此格式的一个使用案例示例是用于解析输出（如堆栈转储），这可能记录在多个条目中。正确模式允许它捕获在单个条目中。

如果同时配置了 `awslogs-datetime-format` 和 `awslogs-multiline-pattern`，此选项始终优先。

> **Note**
>
> 多行日志记录对所有日志消息执行正则表达式解析和匹配，这可能会对日志记录性能产生负面影响。

`awslogs-region`

必需：是

指定 `awslogs` 日志驱动程序应将您的 Docker 日志发送到的区域。您可选择将位于不同区域的集群中的所有日志发送到 CloudWatch Logs 中的单个区域，以便它们全部显示在一个位置，或者您可以按区域将它们分开以更好地粒化。确保指定的日志组存在于您使用此选项指定的区域中。

`awslogs-group`

必需：是

您必须指定 `awslogs` 日志驱动程序将其日志流发送到的日志组。有关更多信息，请参阅创建您的日志组 (p. 98)。

`awslogs-multiline-pattern`

必需：否

此选项使用正则表达式定义多行开始位置模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。因此，匹配行是日志消息之间的分隔符。

如果还配置了 `awslogs-datetime-format`，则会忽略此选项。

> **Note**
>
> 多行日志记录对所有日志消息执行正则表达式解析和匹配。这可能会对日志记录性能产生负面影响。

`awslogs-stream-prefix`

必需：对 EC2 启动类型来说是可选的，对 Fargate 启动类型来说是必需的。

利用 `awslogs-stream-prefix` 选项，可将日志流与指定的前缀、容器名称和容器所属的 Amazon ECS 任务的 ID 关联在一起。如果您使用此选项指定前缀，则日志流将采用以下格式：

```
prefix-name/container-name/ecs-task-id
```

如果您未使用此选项指定前缀，则日志流将以 Docker 守护程序在容器实例上分配的容器 ID 命名。由于仅使用 Docker 容器 ID（仅在容器实例上可用）很难将日志回溯到发送它们的容器，因此，我们建议您使用此选项指定前缀。

对于 Amazon ECS 服务，您可使用服务名称作为前缀，从而允许您将日志流跟踪到容器所属的服务、发送它们的容器的名称以及容器所属的任务的 ID。

您必须为日志指定流前缀，才能在使用 Amazon ECS 控制台时在日志窗格中显示日志。

在任务定义中指定日志配置

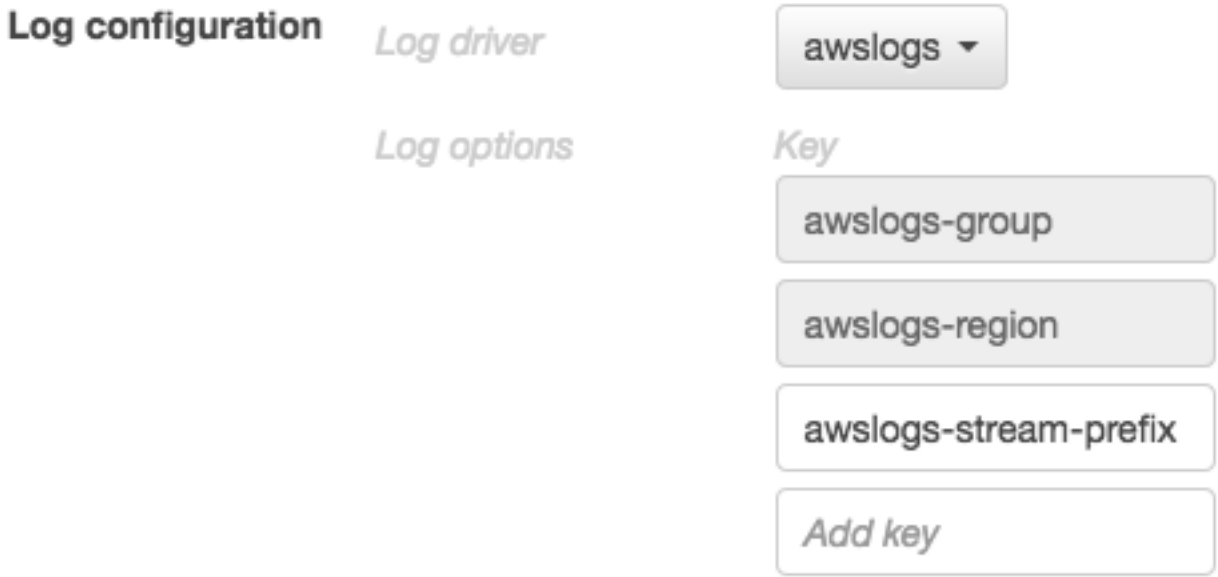
必须先在任务定义中为容器指定 `awslogs` 日志驱动程序，容器才能将日志发送到 CloudWatch。此部分描述了为使用 `awslogs` 日志驱动程序而进行的容器日志配置。有关更多信息，请参阅 创建任务定义 (p. 45)。

下面显示的任务定义 JSON 为每个容器都指定了一个 `logConfiguration` 对象，即，为将日志发送到名为 `awslogs-wordpress` 的日志组的 WordPress 容器指定了一个，并为将日志发送到名为 `awslogs-mysql` 的日志组的 MySQL 容器指定了一个。两个容器都使用 `awslogs-example` 日志流前缀。

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
        {
            "environment": [
                {
                    "name": "MYSQL_ROOT_PASSWORD",
                    "value": "password"
                }
            ],
            "name": "mysql",
            "image": "mysql",
            "cpu": 10,
            "memory": 500,
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            }
        }
    ],
```

```
    "family": "awslogs-example"
}
```

在 Amazon ECS 控制台中，指定了 `wordpress` 容器的日志配置，如下图所示。



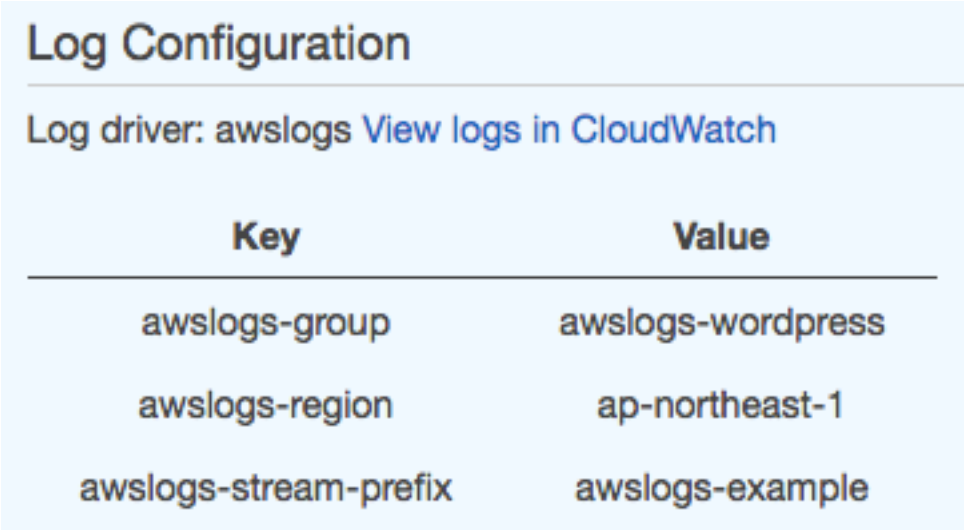
在容器定义日志配置中向 `awslogs` 日志驱动程序注册任务定义之后，可运行任务或使用此任务定义创建服务以开始将日志发送到 CloudWatch Logs。有关更多信息，请参阅 正在运行的任务 (p. 251) 和 创建服务 (p. 317)。

在 CloudWatch Logs 中查看 awslogs 容器日志

对于使用 EC2 启动类型的任务，在容器实例角色有适当的权限将日志发送到 CloudWatch Logs、容器代理最低更新到版本 1.9.0 且您已配置并启动包含使用 `awslogs` 日志驱动程序的容器的任务之后，您配置的容器应会将其日志数据发送到 CloudWatch Logs。您可在控制台中查看和搜索这些日志。

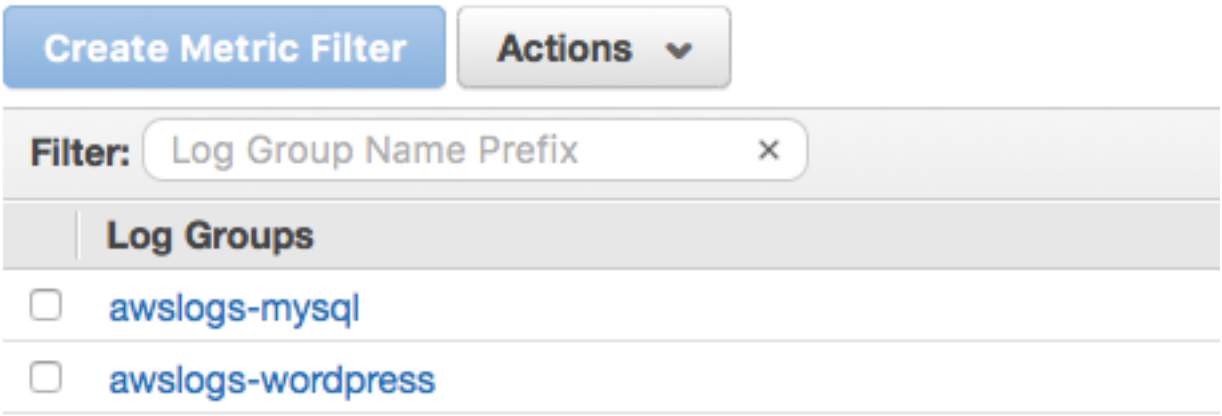
通过 Amazon ECS 控制台查看容器的 CloudWatch Logs 数据

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters 页上，选择包含要查看的任务的集群。
3. 在 Cluster: ***`cluster_name`*** 页上，选择 Tasks 并选择要查看的任务。
4. 在 Task: ***`task_id`*** 页上，通过选择容器名称左侧的箭头来扩展容器视图。
5. 在 Log Configuration (日志配置) 部分中，选择 View logs in CloudWatch (查看 CloudWatch 中的日志)，这将在 CloudWatch 控制台中打开关联的日志流。

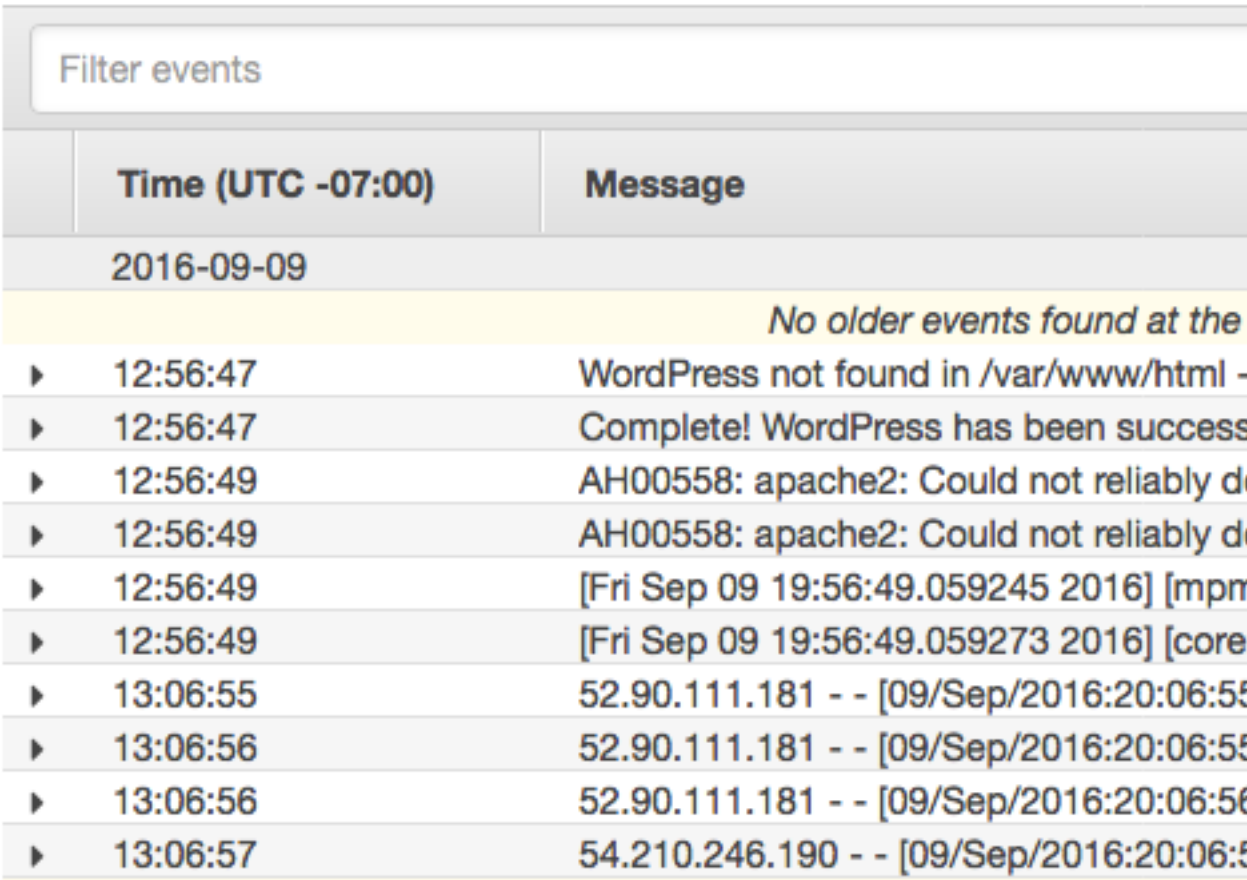


在 CloudWatch 控制台中查看 CloudWatch Logs 数据

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，选择 Logs。
3. 选择要查看的日志组。您应该会看到在创建您的日志组 (p. 98)中创建的日志组。



4. 选择要查看的日志流。



任务的私有注册表身份验证

通过使用 AWS Secrets Manager 对任务进行私有注册表身份验证，您可以在容器定义中安全地存储并随后引用您的凭证。这样，您的任务就可以使用私有存储库中的映像。使用 Fargate 或 EC2 启动类型的任务支持此功能。

> **Important**
>
> 如果您的任务定义引用了存储在 Amazon ECR 中的图像，则此主题不适用。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南 中的将 Amazon ECR 映像与 Amazon ECS 结合使用。

对于使用 EC2 启动类型的任务，此功能需要版本 1.19.0 或更高版本的容器代理；但我们建议您使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

对于使用 Fargate 启动类型的任务，此功能需要平台版本 1.2.0 或更高版本。有关信息，请参阅 AWS Fargate 平台版本 (p. 33)。

在容器定义中，使用您创建的密钥的完整 ARN 指定 `repositoryCredentials`。您引用的密钥可以来自与使用此密钥的任务不同的区域，但必须来自同一个账户。

> **Note**
>
> 使用 Amazon ECS API、AWS CLI 或 AWS 开发工具包时，如果密钥存在于要启动的任务所在的区域，可以使用密钥的完整 ARN 或名称。使用 AWS 管理控制台时，必须指定密钥的完整 ARN。

下面是显示必需参数的任务定义代码段：

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

> **Note**
>
> 启用私有注册表身份验证的另一种方法使用 Amazon ECS 容器代理环境变量向私有注册表进行身份验证。只有使用 EC2 启动类型的任务才支持此方法。有关更多信息，请参阅容器实例的私有注册表身份验证 (p. 228)。

私有注册表身份验证所需的 IAM 权限

使用此功能需要 Amazon ECS 任务执行角色。这允许容器代理拉取容器映像。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

要提供对您创建的密钥的访问权限，请将以下权限作为内联策略手动添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM 策略。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` — 仅当密钥使用自定义 KMS 密钥而不是默认密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

添加权限的示例内联策略如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
        "arn:aws:kms:<region>:<aws_account_id>:key:key_id"
      ]
    }
  ]
}
```

启用私有注册表身份验证

创建基本密钥

使用 AWS Secrets Manager 为您的私有注册表凭证创建密钥。

1. 通过 https://console.aws.amazon.com/secretsmanager/ 打开 AWS Secrets Manager 控制台。

2. 选择存储新密钥。
3. 对于选择密钥类型，选择其他密钥类型。
4. 选择纯文本文件并使用以下格式输入您的私有注册表凭证：

```
{
  "username" : "privateRegistryUsername",
  "password" : "privateRegistryPassword"
}
```

5. 选择 Next (下一步)。
6. 对于密钥名称，请键入可选的路径和名称，如 **`production/MyAwesomeAppSecret`** 或 **`development/TestSecret`**，然后选择下一步。您可以选择添加描述以帮助记住该密钥以后的用途。

   密钥名称应仅包含 ASCII 字母、数字或以下任意字符：/_+=.@-
7. (可选) 此时，您可以为密钥配置轮换。对于此程序，请将其保留为禁用自动轮换，然后选择下一步。

   有关如何为新的或现有的密钥配置轮换的信息，请参阅轮换 AWS Secrets Manager 密钥。
8. 检查您的设置，然后选择 Store secret (存储密钥) 以将输入的所有内容作为新密钥保存在 Secrets Manager 中。

创建使用私有注册表身份验证的任务定义

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Task Definitions。
3. 在 Task Definitions (任务定义) 页面上，选择 Create new Task Definition (创建新任务定义)。
4. 在 Select launch type compatibility (选择启动类型兼容性) 页面上，选择任务的启动类型，然后选择 Next step (下一步)。

   > **Note**
   >
   > 此步骤仅适用于当前支持使用 AWS Fargate 的 Amazon ECS 的区域。有关更多信息，请参阅Amazon ECS 上的 AWS Fargate (p. 28)。
5. 在 Task Definition Name (任务定义名称) 中，为任务定义键入名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
6. 对于 Task execution role (任务执行角色)，选择现有任务执行角色，或者选择 Create new role (创建新角色) 为您创建一个此类角色。此角色授权 Amazon ECS 提取任务的私有映像。有关更多信息，请参阅私有注册表身份验证所需的 IAM 权限 (p. 105)。

   > **Important**
   >
   > 如果 Task execution role (任务执行角色) 字段未显示，请选择 Configure via JSON (通过 JSON 配置) 并手动添加 `executionRoleArn` 字段来指定您的任务执行角色。下面显示语法：
   >
   > ```
   > "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole"
   > ```
7. 对于您的任务定义中要创建的每个容器，请完成以下步骤：

   a. 在 Container Definitions (容器定义) 部分，选择 Add container (添加容器)。
   b. 对于 Container name (容器名称)，键入容器的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
   c. 对于 Image (映像)，键入映像名称或到您私有映像的路径。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
   d. 选择 Private repository authentication (私有存储库身份验证) 选项。
   e. 对于 Secrets manager ARN (密钥管理器 ARN)，输入您之前创建的密钥的完整 Amazon 资源名称 (ARN)。值的长度必须介于 20 到 2048 个字符之间。

f. 填写剩余的必填字段和要在您的容器定义中使用的任何可选字段。更多容器定义参数将可用于 Advanced container configuration (高级容器配置) 菜单中。有关更多信息，请参阅任务定义参数 (p. 52)。

g. 选择 Add。

8. 当添加您的容器时，请选择 Create (创建)。

指定敏感数据

Amazon ECS 使您能够向容器中注入敏感数据，方法是将您的敏感数据存储在 AWS Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数中，然后在容器定义中引用它们。使用 EC2 和 Fargate 启动类型的任务支持此功能。

可以按以下方式将密钥对容器开放：

- 要将敏感数据作为环境变量注入容器，请使用 `secrets` 容器定义参数。
- 要引用容器的日志配置中的敏感信息，请使用 `secretOptions` 容器定义参数。

有关指定敏感数据的注意事项

在指定容器的敏感数据时，应考虑以下事项：

- 对于使用 Fargate 启动类型的任务，此功能要求您的任务使用平台版本 1.3.0 或更高版本。有关信息，请参阅 AWS Fargate 平台版本 (p. 33)。
- 对于使用 EC2 启动类型的任务，此功能要求您的容器实例具有 1.22.0 版或更高版本的容器代理。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。
- 最初启动容器时，会将敏感数据注入容器中。如果随后更新或轮换密钥或 Parameter Store 参数，则容器将不会自动接收已更新的值。您必须启动新任务，或者如果您的任务是服务的一部分，则可以更新服务并使用强制新部署选项来强制服务启动新任务。
- 此功能在 GovCloud（美国东部）区域中不可用。

作为环境变量注入敏感数据

在容器定义中，使用要在容器中设置的环境变量的名称以及包含要提供给容器的敏感数据的 Secrets Manager 密钥或 Systems Manager Parameter Store 参数的完整 ARN 指定 `secrets`。您引用的参数必须来自同一个账户，但可以来自与使用此参数的容器不同的区域。

> **Important**
>
> 如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

有关创建 Secrets Manager 密钥并将其作为环境变量注入容器中的完整教程，请参阅教程：使用 Secrets Manager 密钥指定敏感数据 (p. 557)。

以下是任务定义的片段，其中显示引用 Secrets Manager 密钥时的格式。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
```

```
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-AbCdEf"
    }]
  }]
}
```

以下是任务定义的片段，其中显示引用 Systems Manager Parameter Store 参数时的格式。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

注入日志配置中的敏感数据

在容器定义中，当指定 `logConfiguration` 时，您可以使用要在容器中设置的日志驱动程序选项的名称以及包含要提供给容器的敏感数据的 Secrets Manager 密钥或 Systems Manager Parameter Store 参数的完整 ARN 指定 `secretOptions`。您引用的参数必须来自同一个账户，但可以来自与使用此参数的容器不同的区域。

> **Important**
>
> 如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

以下是任务定义的片段，其中显示引用 Secrets Manager 密钥时的格式。

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://cloud.splunk.com:8080"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

以下是任务定义的片段，其中显示引用 Systems Manager Parameter Store 参数时的格式。

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:parameter_name"
      }]
    }]
```

```
  }]
}
```

Amazon ECS 密钥所需的 IAM 权限

要使用此功能，您必须具有 Amazon ECS 任务执行角色，并在任务定义中引用它。这允许容器代理提取必要的 AWS Systems Manager 或 Secrets Manager 资源。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

要提供对您创建的 AWS Systems Manager Parameter Store 参数的访问权限，请将以下权限作为内联策略手动添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM 策略。

- `ssm:GetParameters` — 当您在任务定义中引用 Systems Manager Parameter Store 参数时是必需的。
- `secretsmanager:GetSecretValue` — 当您直接引用 Secrets Manager 密钥或者您的 Systems Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必需的。
- `kms:Decrypt` — 仅当您的密钥使用自定义 KMS 密钥而不是默认密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

以下示例内联策略添加所需权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/parameter_name",
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
        "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
      ]
    }
  ]
}
```

创建 AWS Secrets Manager 密钥

您可以使用 Secrets Manager 控制台为您的敏感数据创建密钥。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的创建基本密钥。

创建基本密钥

使用 Secrets Manager 为您的敏感数据创建密钥。

1. 通过 https://console.aws.amazon.com/secretsmanager/ 打开 Secrets Manager 控制台。
2. 选择存储新密钥。
3. 对于选择密钥类型，选择其他密钥类型。
4. 以 Key (键) 和 Value (值) 对的形式指定自定义密钥的详细信息。例如，您可以指定键 UserName，然后提供适当的用户名作为其值。添加名为 Password 的第二个键并将密码文本作为其值。您还可以为数据库名称、服务器地址、TCP 端口等添加条目。您可以添加所需数量的对以存储所需的信息。

   或者，您也可以选择 Plaintext (明文) 选项卡，并以所需的任何方式输入密钥值。

5. 选择要用于加密密钥中的受保护文本的 AWS KMS 加密密钥。如果您没有选择一个，则 Secrets Manager 会检查账户是否存在默认密钥并在存在时使用它。如果不存在默认密钥，则 Secrets Manager 将自动为您创建一个。您也可以选择添加新密钥以创建专用于该密钥的自定义 CMK。要创建您自己的 AWS KMS CMK，您必须具有在您的账户中创建 CMK 的权限。
6. 选择 Next (下一步)。
7. 对于密钥名称，请键入可选的路径和名称，如 **`production/MyAwesomeAppSecret`** 或 **`development/TestSecret`**，然后选择下一步。您可以选择添加描述以帮助记住该密钥以后的用途。

   密钥名称应仅包含 ASCII 字母、数字或以下任意字符：/_+=.@-
8. (可选) 此时，您可以为密钥配置轮换。对于此程序，请将其保留为禁用自动轮换，然后选择下一步。

   有关如何为新的或现有的密钥配置轮换的信息，请参阅轮换 AWS Secrets Manager 密钥。
9. 检查您的设置，然后选择 Store secret (存储密钥) 以将输入的所有内容作为新密钥保存在 Secrets Manager 中。

创建 AWS Systems Manager Parameter Store 参数

您可以使用 AWS Systems Manager 控制台为您的敏感数据创建 Systems Manager Parameter Store 参数。有关更多信息，请参阅 AWS Systems Manager 用户指南 中的演练：在命令中创建和使用参数（控制台）。

创建 Parameter Store 参数

1. 在 https://console.aws.amazon.com/systems-manager/ 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，依次选择 Parameter Store 和 Create parameter (创建参数)。
3. 对于 Name (名称)，键入层次结构和参数名称。例如，键入 `test/database_password`。
4. 对于 Description (描述)，键入可选描述。
5. 对于 Type，选择 String、StringList 或 SecureString。

   Note

   - 如果选择 SecureString，则会显示 KMS Key ID (KMS 密钥 ID) 字段。如果您没有提供 KMS CMK ID、KMS CMK ARN、别名或别名 ARN，则系统将使用 `alias/aws/ssm`，这是 Systems Manager 的默认 KMS CMK。要避免使用此密钥，请选择自定义密钥。有关更多信息，请参阅 AWS Systems Manager 用户指南 中的使用安全字符串参数。
   - 在控制台中使用具有自定义 KMS CMK 别名或别名 ARN 的 `key-id` 参数创建安全字符串参数时，您必须在别名前面指定前缀 `alias/`。以下是 ARN 示例：

     ```
     arn:aws:kms:us-east-2:123456789012:alias/MyAliasName
     ```

     以下是别名示例：

     ```
     alias/MyAliasName
     ```
6. 对于 Value (值)，键入一个值。例如：`MyFirstParameter`。如果您选择了 SecureString，则在您键入时值会被掩蔽。
7. 选择创建参数。

创建引用密钥的任务定义

您可以使用 Amazon ECS 控制台创建引用 Secrets Manager 密钥或 Systems Manager Parameter Store 参数的任务定义。

创建指定密钥的任务定义

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，依次选择 Task Definitions (任务定义) 和 Create new Task Definition (创建新任务定义)。
3. 在 Select launch type compatibility (选择启动类型兼容性) 页面上，选择任务的启动类型，然后选择 Next step (下一步)。

   Note

   此步骤仅适用于当前支持使用 AWS Fargate 的 Amazon ECS 的区域。有关更多信息，请参阅Amazon ECS 上的 AWS Fargate (p. 28)。
4. 在 Task Definition Name (任务定义名称) 中，为任务定义键入名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
5. 对于 Task execution role (任务执行角色)，选择现有任务执行角色，或者选择 Create new role (创建新角色) 为您创建一个此类角色。此角色授权 Amazon ECS 提取任务的私有映像。有关更多信息，请参阅私有注册表身份验证所需的 IAM 权限 (p. 105)。

   Important

   如果 Task execution role (任务执行角色) 字段未显示，请选择 Configure via JSON (通过 JSON 配置) 并手动添加 `executionRoleArn` 字段来指定您的任务执行角色。以下代码显示语法：

   ```
   "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole"
   ```
6. 对于您的任务定义中要创建的每个容器，请完成以下步骤：
   a. 在 Container Definitions (容器定义) 下，选择 Add container (添加容器)。
   b. 对于 Container name (容器名称)，键入容器的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
   c. 对于 Image (映像)，键入映像名称或到您私有映像的路径。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
   d. 展开 Advanced container configuration (高级容器配置)。
   e. 对于作为环境变量引用的容器密钥，在 Environment (环境) 下，对于 Environment variables (环境变量)，请完成以下字段：
      i. 对于 Key (键)，输入要在容器中设置的环境变量的名称。此选项对应于容器定义的 `secrets` 部分中的 `name` 字段。
      ii. 对于 Value (值)，选择 ValueFrom。对于添加值，请输入 Secrets Manager 密钥的完整 ARN 或 AWS Systems Manager Parameter Store 参数的名称或完整 ARN，其中包含要作为环境变量提供给您的容器的数据。

      Note

      如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用密钥的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。
   f. 对于在容器的日志配置中引用的密钥，在 Storage and Logging (存储和日志记录) 下，对于 Log configuration (日志配置)，填写以下字段：
      i. 清除自动配置 CloudWatch Logs 选项。
      ii. 在 Log options (日志选项) 下，对于 Key (密钥)，输入要设置的日志配置选项。
      iii. 对于 Value (值)，选择 ValueFrom。对于添加值，请输入 Secrets Manager 密钥的完整 ARN 或 AWS Systems Manager Parameter Store 参数的名称或完整 ARN，其中包含要作为日志选项提供给日志配置的数据。

Note

如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用密钥的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

g. 填写剩余的必填字段和要在您的容器定义中使用的任何可选字段。更多容器定义参数将可用于 Advanced container configuration (高级容器配置) 菜单中。有关更多信息，请参阅任务定义参数 (p. 52)。

h. 选择 Add。

7. 当添加您的容器时，请选择 Create (创建)。

示例任务定义

此部分提供了一些任务定义示例，您可以使用这些示例来开始创建自己的任务定义。有关更多信息，请参阅任务定义参数 (p. 52)和创建任务定义 (p. 45)。

主题

- 示例：Web 服务器 (p. 112)
- 示例：WordPress 和 MySQL (p. 113)
- 示例：awslogs 日志驱动程序 (p. 114)
- 示例：splunk 日志驱动程序 (p. 114)
- 示例：fluentd 日志驱动程序 (p. 115)
- 示例：gelf 日志驱动程序 (p. 115)
- 示例：Amazon ECR 映像和任务定义 IAM 角色 (p. 116)
- 示例：带命令的入口点 (p. 116)
- 示例：容器依赖项 (p. 116)

示例：Web 服务器

下面是用于设置 Web 服务器的示例任务定义 (使用 Fargate 启动类型)：

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!
</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></
html>' >  /usr/local/apache2/htdocs/index.html && httpd-foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
```

```
        },
        "name": "sample-fargate-app",
        "portMappings": [
           {
              "containerPort": 80,
              "hostPort": 80,
              "protocol": "tcp"
           }
        ]
     }
  ],
  "cpu": "256",
  "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
  "family": "fargate-task-definition",
  "memory": "512",
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
      "FARGATE"
   ]
}
```

示例：WordPress 和 MySQL

以下示例指定链接在一起的 WordPress 容器和 MySQL 容器。此 WordPress 容器在主机端口 80 上开放容器端口 80。容器实例上的安全组将需要打开端口 80 以便能够从 Web 浏览器访问此 WordPress 安装。

有关 WordPress 容器的更多信息，请参阅 https://registry.hub.docker.com/_/wordpress/ 上的正式 WordPress Docker Hub 存储库。有关 MySQL 容器的更多信息，请转到 https://registry.hub.docker.com/_/mysql/ 上的正式 MySQL Docker Hub 存储库。

```
{
  "containerDefinitions": [
    {
      "name": "wordpress",
      "links": [
        "mysql"
      ],
      "image": "wordpress",
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "memory": 500,
      "cpu": 10
    },
    {
      "environment": [
        {
          "name": "MYSQL_ROOT_PASSWORD",
          "value": "password"
        }
      ],
      "name": "mysql",
      "image": "mysql",
      "cpu": 10,
      "memory": 500,
      "essential": true
    }
  ],
  "family": "hello_world"
```

```
}
```

> **Important**
>
> 如果您将此任务定义与负载均衡器一起使用，则需要在容器启动后立即通过容器实例上的 Web 接口完成 WordPress 安装程序安装。负载均衡器运行状况检查 ping 将预计来自服务器的 200 响应，但 WordPress 将返回 301，直到安装完成。如果未通过负载均衡器运行状况检查，则负载均衡器会取消注册该实例。

示例：awslogs 日志驱动程序

以下示例演示如何在使用 Fargate 启动类型的任务定义中使用 awslogs 日志驱动程序。nginx 容器将其日志发送到 ecs-log-streaming 区域中的 us-west-2 日志组。有关更多信息，请参阅 使用 awslogs 日志驱动程序 (p. 97)。

```
{
  "containerDefinitions": [
    {
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "name": "nginx-container",
      "image": "nginx",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "ecs-log-streaming",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "fargate-task-1"
        }
      },
      "cpu": 0
    }
  ],
  "networkMode": "awsvpc",
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "memory": "2048",
  "cpu": "1024",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "family": "example_task_1"
}
```

示例：splunk 日志驱动程序

以下示例演示如何在任务定义中使用 splunk 日志驱动程序，以将日志发送到远程服务。Splunk 令牌参数指定为密钥选项，因为它可能被视为敏感数据。有关更多信息，请参阅 指定敏感数据 (p. 107)。

```
"containerDefinitions": [{
  "logConfiguration": {
   "logDriver": "splunk",
   "options": {
    "splunk-url": "https://cloud.splunk.com:8080",
    "tag": "tag_name",
```

```
   },
   "secretOptions": [{
    "name": "splunk-token",
    "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:splunk-token-KnrBkD"
}],
```

示例：fluentd 日志驱动程序

以下示例演示如何在任务定义中使用 fluentd 日志驱动程序，以将日志发送到远程服务。fluentd-address 值被指定为密钥选项，因为它可能会视为敏感数据。有关更多信息，请参阅 指定敏感数据 (p. 107)。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "fluentd",
  "options": {
   "tag": "fluentd demo"
  },
  "secretOptions": [{
   "name": "fluentd-address",
   "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:fluentd-address-
KnrBkD"
  }]
 },
 "entryPoint": [],
 "portMappings": [{
             "hostPort": 80,
             "protocol": "tcp",
             "containerPort": 80
             },
             {
  "hostPort": 24224,
  "protocol": "tcp",
  "containerPort": 24224
 }]
}],
```

示例：gelf 日志驱动程序

以下示例演示如何在任务定义中使用 gelf 日志驱动程序，以将日志发送到远程主机（此主机运行将 Gelf 日志作为输入的 Logstash）。有关更多信息，请参阅 logConfiguration (p. 64)。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "gelf",
  "options": {
   "gelf-address": "udp://logstash-service-address:5000",
   "tag": "gelf task demo"
  }
 },
 "entryPoint": [],
 "portMappings": [{
   "hostPort": 5000,
   "protocol": "udp",
   "containerPort": 5000
  },
  {
   "hostPort": 5000,
   "protocol": "tcp",
   "containerPort": 5000
  }
```

```
 ]
}],
```

示例：Amazon ECR 映像和任务定义 IAM 角色

以下示例使用 `123456789012.dkr.ecr.us-west-2.amazonaws.com` 注册表中带 `v1` 标记的名为 `aws-nodejs-sample` 的 Amazon ECR 映像。此任务中的容器继承来自 `arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole` 角色的 IAM 权限。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

```
{
  "containerDefinitions": [
    {
      "name": "sample-app",
      "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/aws-nodejs-sample:v1",
      "memory": 200,
      "cpu": 10,
      "essential": true
    }
  ],
  "family": "example_task_3",
  "taskRoleArn": "arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole"
}
```

示例：带命令的入口点

以下示例演示使用入口点和命令参数的 Docker 容器的语法。此容器将对 `google.com` 执行四次 ping 操作，然后退出。

```
{
  "containerDefinitions": [
    {
      "memory": 32,
      "essential": true,
      "entryPoint": [
        "ping"
      ],
      "name": "alpine_ping",
      "readonlyRootFilesystem": true,
      "image": "alpine:3.4",
      "command": [
        "-c",
        "4",
        "google.com"
      ],
      "cpu": 16
    }
  ],
  "family": "example_task_2"
}
```

示例：容器依赖项

此示例展示了具有多个容器（其中指定了容器依赖项）的任务定义的语法。在以下任务定义中，`envoy` 容器必须达到正常运行状态（由必需的容器运行状况检查参数决定），然后 `app` 容器才能启动。有关更多信息，请参阅容器依赖项 (p. 71)。

```
{
```

```
  "family": "appmesh-gateway",
  "proxyConfiguration":{
      "type": "APPMESH",
      "containerName": "envoy",
      "properties": [
          {
              "name": "IgnoredUID",
              "value": "1337"
          },
          {
              "name": "ProxyIngressPort",
              "value": "15000"
          },
          {
              "name": "ProxyEgressPort",
              "value": "15001"
          },
          {
              "name": "AppPorts",
              "value": "9080"
          },
          {
              "name": "EgressIgnoredIPs",
              "value": "169.254.170.2,169.254.169.254"
          }
      ]
  },
  "containerDefinitions": [
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-
prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
          "name": "ENVOY_LOG_LEVEL",
          "value": "info"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "echo hello"
        ],
```

```
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

更新任务定义

要更新任务定义，请创建任务定义修订。如果任务定义用于服务中，则您必须更新该服务才能使用更新的任务定义。

创建任务定义修订

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择包含您的任务定义的区域。
3. 在导航窗格中，选择 Task Definitions。
4. 在 Task Definitions 页面上，选择要修订的任务定义左侧的框，然后选择 Create new revision。
5. 在 Create new revision of Task Definition 页面上，进行更改。例如，要更改现有的容器定义 (如容器映像、内存限制或端口映射)，请选择容器，进行更改，然后选择 Update。
6. 验证信息并选择 Create。
7. 如果您的任务定义用于服务中，请用更新的任务定义更新您的服务。有关更多信息，请参阅 更新服务 (p. 326)。

取消注册任务定义

如果确定 Amazon ECS 中不再需要任务定义，可以取消注册任务定义，这样，如果需要运行任务或更新服务，任务定义将不再显示在 `ListTaskDefinition` API 调用或控制台中。

在取消注册任务定义后，它将立即被标记为 `INACTIVE`。引用 `INACTIVE` 任务定义的现有任务和服务将继续运行而不会出现中断，引用 `INACTIVE` 任务定义的现有服务仍可通过修改服务的预期数目来向上扩展或向下扩展。

您无法使用 `INACTIVE` 任务定义运行新任务或创建新服务，而且无法更新现有服务以引用 `INACTIVE` 任务定义 (尽管在取消注册后的一个长达 10 分钟的时段内，这些限制可能尚未生效)。

> **Note**
>
> 目前，在您的账户中仍可发现 `INACTIVE` 任务定义；然而此行为将来可能会发生变化。因此，不应依赖 `INACTIVE` 任务定义持续保留已超过生命周期的相关任务和服务。

使用以下过程可取消注册任务定义。

取消注册任务定义

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择包含您的任务定义的区域。
3. 在导航窗格中，选择 Task Definitions。
4. 在 Task Definitions 页面上，选择包含要取消注册的一个或多个修订的任务定义名称。
5. 在任务定义名称页面上，选中要取消注册的每个任务定义修订左侧的框。

6. 选择 Actions、Deregister。
7. 验证 Deregister Task Definition 窗口中的信息，然后选择 Deregister 以完成操作。

账户设置

Amazon ECS 提供了以下账户设置，这些设置可让您选择使用或退出特定功能。

Amazon 资源名称 (ARN) 和 IDS

资源名称：`serviceLongArnFormat`、`taskLongArnFormat` 和 `containerInstanceLongArnFormat`

Amazon ECS 正在引入针对 Amazon ECS 服务、任务和容器实例的 Amazon 资源名称 (ARN) 和资源 ID 的新格式。您必须为每种资源类型选择使用新格式，以对该资源类型使用资源标记等功能。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。

AWSVPC 中继

资源名称：`awsvpcTrunking`

Amazon ECS 支持使用受支持的 Amazon EC2 实例类型来启动具有提高的弹性网络接口 (ENI) 密度的容器实例。当您使用这些实例类型并选择使用 `awsvpcTrunking` 账户设置时，其他 ENI 将在新启动的容器实例上可用。此配置允许您使用 `awsvpc` 网络模式在每个容器实例上放置更多任务。通过使用此功能，启用了 `awsvpcTrunking` 的 `c5.large` 实例的 ENI 限制提高了 10。容器实例将具有一个主网络接口，而 Amazon ECS 将创建一个“中继”网络接口并将此接口附加到容器实例。主网络接口或中继网络接口都不会计入 ENI 限制，因此，此配置允许您在容器实例上启动 10 个任务，而不是当前的两个任务。有关更多信息，请参阅弹性网络接口中继 (p. 175)。

CloudWatch Container Insights

资源名称：`containerInsights`

CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合和汇总指标与日志。指标包括资源的使用率，如 CPU、内存、磁盘和网络。Container Insights 还提供诊断信息（如容器重新启动失败），以帮助您查明问题并快速解决问题。您还可以设置 Container Insights 收集的指标的 CloudWatch 警报。有关更多信息，请参阅Amazon ECS CloudWatch Container Insights (p. 363)。

可以选择使用 `containerInsights` 账户设置为所有集群启用 Container Insights，也可以在集群创建期间使用集群设置在单个集群上启用 Container Insights。

对于包含使用 EC2 启动类型的的任务或服务的集群，您的容器实例必须运行版本 1.29.0 或更高版本的 Amazon ECS 代理。有关更多信息，请参阅 Amazon ECS 容器代理版本 (p. 207)。

对于每个区域，您可以在账户级别或为特定的 IAM 用户或角色选择使用或选择不使用每个账户设置。可供选择使用或选择不使用的账户设置包括新的 ARN 和资源 ID 格式以及 awsvpc 中继功能。

下面是支持的方案：

- IAM 用户或角色可以选择使用或选择不使用其个人用户账户。
- IAM 用户或角色可以为账户上的所有用户设置默认的选择使用或选择不使用设置。
- 根用户能够选择使用或选择不使用账户上的任何特定 IAM 角色或用户。如果根用户的账户设置发生更改，则会为未选择单独账户设置的所有 IAM 用户和角色设置默认值。

必须为每个账户设置单独选定选择使用和选择不使用选项。资源的 ARN 和资源 ID 格式由创建了资源的 IAM 用户或角色的选择状态定义。

仅在选择使用后启动的资源才能采用新的 ARN 和资源 ID 格式或增加的 ENI 限制。所有现有资源均不受影响。要使 Amazon ECS 服务和任务转换为新 ARN 和资源 ID 格式，必须重新创建该服务或任务。要将容器

实例转换为新 ARN 和资源 ID 格式或增加的 ENI 限制，必须耗尽该容器实例，然后向集群注册一个新的容器实例。

> **Note**
>
> 对于 Amazon ECS 服务启动的任务，仅在该服务是在 2018 年 11 月 16 日或之后创建的，并且创建该服务的 IAM 用户为任务选择使用新格式时，才能采用新 ARN 和资源 ID 格式。

主题

- Amazon 资源名称 (ARN) 和 ID (p. 121)
- 查看账户设置 (p. 122)
- 修改账户设置 (p. 123)

Amazon 资源名称 (ARN) 和 ID

在创建 Amazon ECS 资源时，将为每个资源分配一个唯一的 Amazon 资源名称 (ARN) 和资源标识符 (ID)。如果您正在通过命令行工具或 Amazon ECS API 使用 Amazon ECS，则某些命令需要资源 ARN 或 ID。例如，如果您正在使用 stop-task AWS CLI 命令来停止任务，则必须在该命令中指定任务 ARN 或 ID。

以下各部分介绍 ARN 和资源 ID 格式如何变化。有关转换为新格式的更多信息，请参阅 Amazon Elastic Container Service 常见问题。

Amazon 资源名称 (ARN) 格式

一些资源具有容易记住的名称，例如名为 `production` 的服务。在其他情况下，您必须使用 Amazon 资源名称 (ARN) 格式指定资源。Amazon ECS 任务、服务和容器实例的新 ARN 格式包括集群名称。有关选择使用新 ARN 格式的详细信息，请参阅 修改账户设置 (p. 123)。

> **Note**
>
> 新 ARN 格式在 GovCloud（美国东部）区域中不可用。

下表显示了每个资源类型的当前（旧）格式和新格式。

资源类型	ARN
容器实例	旧：`arn:aws:ecs:region:aws_account_id:container-instance/container-instance-id` 新：`arn:aws:ecs:region:aws_account_id:container-instance/cluster-name/container-instance-id`
Amazon ECS 服务	旧：`arn:aws:ecs:region:aws_account_id:service/service-name` 新：`arn:aws:ecs:region:aws_account_id:service/cluster-name/service-name`
Amazon ECS 任务	旧：`arn:aws:ecs:region:aws_account_id:task/task-id` 新：`arn:aws:ecs:region:aws_account_id:task/cluster-name/task-id`

资源 ID 长度

资源 ID 采用字母和数字的唯一组合形式。新资源 ID 格式包括 Amazon ECS 任务和容器实例的更短 ID。旧资源 ID 格式的长度为 36 个字符。新 ID 采用 32 个字符的格式，不包含任何连字符。有关选择使用新资源 ID 格式的详细信息，请参阅 修改账户设置 (p. 123)。

> **Note**
>
> 新资源 ID 格式在 GovCloud（美国东部）区域中不可用。

时间表

新格式有一个选择周期。下面是与此更改相关的重要日期。

- 从初始格式发布到 2019 年 3 月 31 日 – 按区域提供选择使用和选择不使用新 Amazon 资源名称 (ARN) 和资源 ID 的功能。创建的任何新账户将默认选择不使用。
- 2019 年 4 月 1 日 - 2019 年 12 月 31 日 – 所有新账户将默认选择使用。将继续按区域提供选择使用和选择不使用的功能。
- 2020 年 1 月 1 日 – 所有账户将默认选择使用。创建的所有新资源将采用新格式。

在选择周期内，您可以随时选择使用或选择不使用新的 Amazon 资源名称 (ARN) 和资源 ID 格式。在选择使用后，您创建的任何新资源都将使用新格式。

> **Note**
>
> 资源 ID 在创建后不会更改。因此，在选择周期内选择使用或不使用新格式不会影响现有的资源 ID。

查看账户设置

您可以使用 AWS 管理控制台和 AWS CLI 工具查看支持新 ARN 和 ID 格式的资源类型或或增加的 ENI 限制。

使用控制台查看账户设置

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在屏幕顶部的导航栏中，选择要查看其账户设置的区域。
3. 从控制面板中选择账户设置。
4. 在 Amazon ECS ARN and resource ID settings (Amazon ECS ARN 和资源 ID 设置)、AWSVPC Trunking (AWSVPC 中继) 和 CloudWatch Container Insights 部分中，您可以查看经过身份验证的 IAM 用户和角色的每个账户设置的选择使用状态。

使用命令行查看账户设置

使用下列命令之一来查看账户设置。

- list-account-settings (AWS CLI)

```
aws ecs list-account-settings --effective-settings --region us-east-1
```

- Get-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

```
Get-ECSAccountSetting -EffectiveSetting true -Region us-east-1
```

使用命令行查看特定 IAM 用户或 IAM 角色的账户设置

使用下列命令之一并在请求中指定 IAM 用户、IAM 角色或根账户用户的 ARN 来查看其账户设置。

- list-account-settings (AWS CLI)

```
aws ecs list-account-settings --principal-arn
 arn:aws:iam::aws_account_id:user/principalName --effective-settings --region us-east-1
```

- Get-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

```
Get-ECSAccountSetting -PrincipalArn arn:aws:iam::aws_account_id:user/principalName -
EffectiveSetting true -Region us-east-1
```

修改账户设置

您可以使用 AWS 管理控制台 和 AWS CLI 工具来修改账户设置。

使用控制台修改账户设置

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在屏幕顶部的导航栏中，选择要修改其账户设置的区域。
3. 从控制面板中选择账户设置。
4. 在 Amazon ECS ARN and resource ID settings (Amazon ECS ARN 和资源 ID 设置)、AWSVPC Trunking (AWSVPC 中继) 和 CloudWatch Container Insights 部分中，您可以选中或取消选中经过身份验证的 IAM 用户和角色的每个账户设置的对应复选框。完成后，选择保存。

   > **Important**
   >
   > IAM 用户和 IAM 角色需要 `ecs:PutAccountSetting` 权限来执行此操作。

5. 在确认屏幕上，选择 Confirm (确认) 以保存选择。

使用命令行修改账户中所有 IAM 用户或角色的默认账户设置

使用下列命令之一修改账户中所有 IAM 用户或角色的默认账户设置。这些更改将应用于整个 AWS 账户，除非一个 IAM 用户或角色显式覆盖自己的这些设置。

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled --
region us-east-2
```

  您也可以使用此命令修改所有任务 (`taskLongArnFormat`)、容器实例 (`containerInstanceLongArnFormat`) 的账户设置，并选择使用容器实例 (`awsvpcTrunking`) 的增加的弹性网络接口 (ENI) 限制。为此，请将 `name` 参数替换为相应的资源类型。

- Write-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-
east-1 -Force
```

使用命令行修改您的 IAM 用户账户的账户设置

使用下列命令之一修改您的 IAM 用户的账户设置。如果您以根用户身份使用这些命令，则更改将应用于整个 AWS 账户，除非一个 IAM 用户或角色显式覆盖自己的这些设置。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --region us-
east-1
```

您也可以使用此命令修改所有任务 (taskLongArnFormat)、容器实例 (containerInstanceLongArnFormat) 的账户设置，并选择使用容器实例 (awsvpcTrunking) 的增加的弹性网络接口 (ENI) 限制。为此，请将 name 参数替换为相应的资源类型。

- Write-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

使用命令行修改特定 IAM 用户或 IAM 角色的账户设置

使用下列命令之一并在请求中指定 IAM 用户、IAM 角色或根账户用户的 ARN 来修改特定 IAM 用户或 IAM 角色的账户设置。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --principal-arn
 arn:aws:iam::aws_account_id:user/principalName --region us-east-1
```

您也可以使用此命令修改所有任务 (taskLongArnFormat)、容器实例 (containerInstanceLongArnFormat) 的账户设置，并选择使用容器实例 (awsvpcTrunking) 的增加的弹性网络接口 (ENI) 限制。为此，请将 name 参数替换为相应的资源类型。

- Write-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

Amazon ECS 容器实例

Amazon ECS 容器实例是一个 Amazon EC2 实例，它运行的是 Amazon ECS 容器代理并且已注册到集群。当您使用 Amazon ECS 运行任务时，使用 EC2 启动类型的任务将放置在活动的容器实例上。

> Note
>
> 由于使用 Fargate 启动类型的任务部署到由 AWS 管理的基础设施上，因此本主题不适用。

主题

- 容器实例概念 (p. 125)
- 容器实例生命周期 (p. 126)
- 检查您的账户的实例角色 (p. 126)
- Amazon ECS-optimized AMI (p. 127)
- 检索 Amazon ECS-Optimized AMI 元数据 (p. 155)
- 订阅Amazon ECS-Optimized Amazon Linux AMI 更新通知 (p. 161)
- 启动 Amazon ECS 容器实例 (p. 165)
- 使用 Amazon EC2 用户数据引导启动容器实例 (p. 168)
- 弹性网络接口中继 (p. 175)
- 连接到您的容器实例 (p. 180)
- 将 CloudWatch Logs 用于容器实例 (p. 181)
- 容器实例耗尽 (p. 190)
- 容器实例内存管理 (p. 190)
- 使用 AWS Systems Manager 远程管理容器实例 (p. 192)
- 在容器实例启动时启动任务 (p. 194)
- 取消注册容器实例 (p. 196)

容器实例概念

- 您的容器实例必须运行 Amazon ECS 容器代理才能注册到您的某个集群中。如果您使用的是Amazon ECS-optimized AMI，则已安装该代理。要使用其他操作系统，请安装代理。有关更多信息，请参阅Amazon ECS 容器代理 (p. 198)。
- 因为 Amazon ECS 容器代理代表您调用 Amazon ECS，所以，必须通过对您的账户进行身份验证并提供所需资源权限的 IAM 角色才能启动容器实例。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色 (p. 404)。
- 如果与您的任务关联的任何容器需要外部连接，您可以将其网络端口映射到主机 Amazon ECS 容器实例上的端口，以便能通过 Internet 访问这些容器。您的容器实例安全组必须允许对您要公开的端口进行入站访问。有关更多信息，请参阅 Amazon VPC 入门指南中的创建安全组。
- 我们强烈建议在 VPC 中启动您的容器实例，因为 Amazon VPC 可提供对您的网络的更多控制并提供更广泛的配置功能。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例） 中的 Amazon EC2 和 Amazon Virtual Private Cloud。

- 容器实例需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的容器实例完成。

  有关接口 VPC 终端节点的更多信息，请参阅Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink) (p. 337)。

  如果您没有配置接口 VPC 终端节点并且您的容器实例没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关和本指南中的 HTTP 代理配置 (p. 245)。有关更多信息，请参阅Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters (p. 555)。
- 为容器实例选择的 Amazon EC2 实例的类型将决定集群中可用的资源。Amazon EC2 提供不同的实例类型，每种类型均具有可用于运行任务的不同的 CPU、内存、存储和网络容量。有关更多信息，请参阅 Amazon EC2 实例。
- 每个容器实例均具有本地存储在容器实例和 Amazon ECS 中的唯一状态信息，
  - 因此，您不应将实例从一个集群取消注册，然后再将其重新注册到另一个集群。要重新定位容器实例资源，建议您从一个集群终止容器实例，然后在新集群中使用最新的Amazon ECS-optimized Amazon Linux 2 AMI 启动新容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例和启动 Amazon ECS 容器实例 (p. 165)。
  - 您无法停止容器实例并更改其实例类型。相反，建议您终止容器实例，然后在所需集群中使用所需实例大小和最新的Amazon ECS-optimized Amazon Linux 2 AMI 启动新容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例和本指南中的启动 Amazon ECS 容器实例 (p. 165)。

容器实例生命周期

当 Amazon ECS 容器代理将实例注册到集群时，容器实例将其状态报告为 `ACTIVE`，并将其代理连接状态报告为 `TRUE`。此容器实例可接受运行任务请求。

如果停止（而不是终止）Amazon ECS 容器实例，则状态将保持为 `ACTIVE`，但代理连接状态将在几分钟内转换为 `FALSE`。在容器实例上运行的任何任务将停止。如果您再次启动容器实例，则容器代理将使用 Amazon ECS 服务重新连接，并且您能够在实例上再次运行任务。

> **Important**
>
> 如果您停止并启动某个容器实例，即重启该实例，则一些旧版本的 Amazon ECS 容器代理会在未取消注册原始容器实例 ID 的情况下再次注册该实例。这种情况下，Amazon ECS 会在集群中列出超出实际拥有数量的容器实例。（如果具有同一 Amazon EC2 实例 ID 的重复容器实例 ID，则可以安全地取消注册列出为 `ACTIVE` 且代理连接状态为 `FALSE` 的重复项。）此问题已在当前版本的 Amazon ECS 容器代理中修复。有关更新到当前版本的更多信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

如果将容器实例的状态更改为 `DRAINING`，则新任务不会置于该容器实例中，并将尽可能删除在此容器实例上运行的所有服务任务，以便执行系统更新。有关更多信息，请参阅容器实例耗尽 (p. 190)。

如果您取消注册或终止某个容器实例，则该容器实例状态将立即更改为 `INACTIVE`，而且在您列出您的容器实例时将不再报告该容器实例。但您仍可以在终止后的 1 小时内描述容器实例。1 小时后，该实例描述将不再可用。

检查您的账户的实例角色

Amazon ECS 容器代理代表您调用 Amazon ECS API。运行该代理的容器实例需要一个 IAM 策略和角色，以便该服务了解该代理属于您。

大多数情况下，在控制台首次运行体验中将自动为您创建 Amazon ECS 实例角色。您可使用以下过程检查并确定您的账户是否已有 Amazon ECS 服务角色。

在 IAM 控制台中检查 `ecsInstanceRole`

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsInstanceRole`。如果角色存在，则您无需创建它。如果角色不存在，请按照 Amazon ECS 容器实例 IAM 角色 (p. 404)中的过程操作来创建角色。

Amazon ECS-optimized AMI

基本 Amazon Elastic Container Service (Amazon ECS) 容器实例规范包括以下内容：

必填

- 运行至少 3.10 版 Linux 内核的现代 Linux 分发版。
- Amazon ECS 容器代理（最好是最新版本）。有关更多信息，请参阅 Amazon ECS 容器代理 (p. 198)。
- 运行至少 1.9.0 版的 Docker 守护程序以及任何 Docker 运行时依赖项。有关更多信息，请参阅 Docker 文档中的检查运行时依赖项。

  > **Note**
  >
  > 要获得最佳体验，建议您使用所使用的相应 Amazon ECS 代理版本附带的且经测试的 Docker 版本。有关更多信息，请参阅 Amazon ECS 容器代理版本 (p. 207)。

推荐

- 用于运行和监控 Amazon ECS 代理的初始化和 nanny 流程。Amazon ECS-optimized AMI 使用 `ecs-init` RPM 来管理代理。有关更多信息，请参阅 GitHub 上的 `ecs-init` 项目。

Amazon ECS-optimized AMI 已根据这些要求和建议进行预配置。建议对容器实例使用Amazon ECS-optimized Amazon Linux 2 AMI，除非应用程序要求使用特定的操作系统或在该 AMI 中尚不可用的 Docker 版本。

Amazon ECS 提供了已针对服务进行了优化的 AMI，包括以下变体。

- Amazon ECS-optimized Amazon Linux 2 AMI – 推荐用于在大多数情况下启动您的 Amazon ECS 容器实例。
- Amazon ECS-optimized Amazon Linux 2 (arm64) AMI – 推荐用于在使用 Amazon EC2 A1 实例类型（由基于 Arm 的 AWS Graviton 处理器提供支持）时启动您的 Amazon ECS 容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例） 中的通用实例。
- Amazon ECS GPU-optimized AMI – 推荐用于在使用 GPU 工作负载时启动您的 Amazon ECS 容器实例。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。
- Amazon ECS-optimized Amazon Linux AMI – 此 AMI 基于 Amazon Linux 1。我们建议您将工作负载迁移到 Amazon ECS-optimized Amazon Linux 2 AMI。对 Amazon ECS-optimized Amazon Linux AMI 的支持将截止于 2020 年 6 月 30 日。
- Amazon ECS-optimized Windows 2019 AMI – 推荐用于在 Windows 操作系统上启动您的 Amazon ECS 容器实例。有关更多信息，请参阅 Windows 容器 (p. 599)。
- Amazon ECS-optimized Windows 2016 AMI – 可用于在 Windows 操作系统上启动您的 Amazon ECS 容器实例。有关更多信息，请参阅 Windows 容器 (p. 599)。

尽管您可以自行创建在 Amazon ECS 上运行您的容器化工作负载所需的符合基本规范的容器实例 AMI，但 AWS 工程师仍会在 Amazon ECS 上预配置并测试经 Amazon ECS 优化的 AMI。这是可供您开始操作并快速获取 AWS 上运行的容器的最简单方式。

每个变体的经 Amazon ECS 优化的 AMI 元数据（包括 AMI ID）均可通过编程方式检索。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。

根据您选择的变体在以下选项卡之一上查看 AMI ID。

Amazon Linux 2

当前 Amazon ECS-optimized Amazon Linux 2 AMI 包括：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux 2 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0dca97e7cde7be3d5	Launch instance
us-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0fac5486e4cff37f4	Launch instance
us-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0c6e63b58aac1048e	Launch instance
us-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0e5e051fd0b505db6	Launch instance
ap-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-02252d984c7e3595a	Launch instance
ap-northeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04a735b489d2a0320	Launch instance
ap-northeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0accbb5aa909be7b6	Launch instance
ap-south-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a8bf4e187339e2c1	Launch instance
ap-southeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-05c6d22d98f9747be	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-southeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-039bb4c3a7946ce1a	Launch instance
ca-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-01c07ee95e77abbb8	Launch instance
eu-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0650e7d86452db33b	Launch instance
eu-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-059aa04f0c253adfb	Launch instance
eu-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0ae254c8a2d3346a7	Launch instance
eu-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0de1dc478496a9e2b	Launch instance
eu-west-3	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0d260f3e5ccd06043	Launch instance
sa-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-038707d64e5b8e7ba	Launch instance
us-gov-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a224902b35f8adfe	Launch instance
us-gov-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04c68165	Launch instance
cn-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance
cn-northwest-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance

Amazon Linux 2 (arm64)

当前 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 包括：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-02a59fb754c85ab16	Launch instance
us-east-1	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0b3b892651e52f0b1	Launch instance
us-west-2	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0860edcdc9c9533e2	Launch instance
eu-west-1	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0af65bdd9a59a3171	Launch instance

Amazon Linux 2 (GPU)

当前 Amazon ECS GPU-optimized AMI 包含以下内容：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)
- 建议的 NVIDIA 驱动程序版本 (`418.40.04`)
- NVIDIA 容器运行时挂钩版本 (`v1.4.0-1`)
- 建议的 CUDA 版本 (`9.2.88`)

下表按区域列出了当前的 Amazon ECS GPU-optimized AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0522cd69e8a331c3a	Launch instance
us-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0875a1e6c2db1cdc9	Launch instance
us-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0dfe31c7ed6577eb6	Launch instance
us-west-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0bf12263037af5756	Launch instance
ap-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0efcda0accbf5a0f4	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-northeast-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-041843b72dde36df4	Launch instance
ap-northeast-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0e18b01ac7608adc1	Launch instance
ap-south-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0c5cdf93d467bec38	Launch instance
ap-southeast-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-07b25dd477ea57f32	Launch instance
ap-southeast-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-07f990e4e7551b774	Launch instance
ca-central-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-08e1aab6a2cfd87e1	Launch instance
eu-central-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-096ccf1edd62587be	Launch instance
eu-north-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0576489203c1c4cce	Launch instance
eu-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-03867df65b0e5abb2	Launch instance
eu-west-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0faecf24ba37edfce	Launch instance
eu-west-3	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0dec5973b167d5ff3	Launch instance
sa-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0a39851053709497a	Launch instance
us-gov-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0365a6c905c4ba3b1	Launch instance
us-gov-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-f8b8ff99	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
cn-north-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-01143a340285bc1fb	Launch instance
cn-northwest-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0240ef2b16197fef3	Launch instance

Amazon Linux AMI

当前 Amazon ECS-optimized Amazon Linux AMI 包括：

- 最新的最简版本的 Amazon Linux AMI
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0151b45908571e14e	Launch instance
us-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0c09d65d2051adab3	Launch instance
us-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0b4f28359911d5826	Launch instance
us-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-077368b501184adb9	Launch instance
ap-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0eeaf74d330e7e8ca	Launch instance
ap-northeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-084d4626fec6f3cb3	Launch instance
ap-northeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-03c5d1f1ba6ec1b7d	Launch instance
ap-south-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-09f6bc049f323302e	Launch instance
ap-southeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-01813416d767d88db	Launch instance
ap-southeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-07fcafbb9f3a4f7db	Launch instance
ca-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-011742239f8f42f46	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
eu-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0bceb1887b6b37130	Launch instance
eu-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0d7012b0cae33c045	Launch instance
eu-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-02b9d5ef54d57fb7d	Launch instance
eu-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0f49b2a901463508a	Launch instance
eu-west-3	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0db8d6e7dcf3cb362	Launch instance
sa-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0ad0be326813ddab6	Launch instance
us-gov-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-033b28d8ef5fe3f72	Launch instance
us-gov-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-b0bafdd1	Launch instance
cn-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-041f913b515304af9	Launch instance
cn-northwest-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0be42f3b41b17b512	Launch instance

Windows Server 2019

当前 Amazon ECS-optimized Windows 2019 AMI 包括：

- 最新版本的 Microsoft Windows Server 2019
- Docker EE 版本 `18.09.8`
- Amazon ECS 容器代理版本 `1.29.0`

下表按区域列出了当前的 Amazon ECS-optimized Windows 2019 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-05c650b3e1759293a	Launch instance
us-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-00f3f16f1d05e53a9	Launch instance
us-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-08f56c127cc9ae9c4	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
us-west-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0c37f25e59d752cb1	Launch instance
ap-northeast-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0771bfb73bc362295	Launch instance
ap-northeast-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0849ba95174008d74	Launch instance
ap-south-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0dfe1bb9ed2dd2ff1	Launch instance
ap-southeast-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-06daa78e84b80091a	Launch instance
ap-southeast-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-071e43d09ba6f2479	Launch instance
ca-central-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0ed093f8cd0bb6f64	Launch instance
eu-central-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-020bded18f4df2c7d	Launch instance
eu-north-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0269f4d7fa129ddcb	Launch instance
eu-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e3604f46f9ccc506	Launch instance
eu-west-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0746c037f8889f6e8	Launch instance
eu-west-3	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-022ae3cb06a066371	Launch instance
sa-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-094ff1452e5849bf6	Launch instance
us-gov-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e238cac0ef640942	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
us-gov-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-13fcbd72	Launch instance
cn-north-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-02728cc741b470390	Launch instance
cn-northwest-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e1cafc0e9ae3faf8	Launch instance

Windows Server 2016

当前 Amazon ECS-optimized Windows 2016 AMI 包括：

- 最新版本的 Microsoft Windows Server 2016
- Docker EE 版本 `18.09.8`
- Amazon ECS 容器代理版本 `1.29.0`

下表按区域列出了当前的 Amazon ECS-optimized Windows 2016 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0ca3903548e4a8631	Launch instance
us-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0831d21a034cb33e4	Launch instance
us-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-08b04db86cc171254	Launch instance
us-west-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0b380fdc264da2fc3	Launch instance
ap-northeast-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0d3f659d5bab5c8e5	Launch instance
ap-northeast-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0d4fe34a4a5cb98ba	Launch instance
ap-south-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0add8558b7afe4950	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-southeast-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-001b756da3bc277be	Launch instance
ap-southeast-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-063f16052f2b588a2	Launch instance
ca-central-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-07cc1c48c86d988ca	Launch instance
eu-central-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0644eb70b836aa5ba	Launch instance
eu-north-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-038a1a6d22e486228	Launch instance
eu-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0e41be650397b1316	Launch instance
eu-west-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0884e795ec651104a	Launch instance
eu-west-3	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-03f921d23a15ff7b1	Launch instance
sa-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-06955dc02fc2a3b7b	Launch instance
us-gov-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-04c0f4fe213678f2b	Launch instance
us-gov-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-38fdbc59	Launch instance
cn-north-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-010da5256208bcb66	Launch instance
cn-northwest-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0c1033d4d4607074f	Launch instance

主题

- 如何启动最新的 Amazon ECS-Optimized Amazon Linux AMI (p. 137)
- 经 Amazon ECS 优化的 AMI 版本 (p. 146)
- AMI 存储配置 (p. 151)

如何启动最新的 Amazon ECS-Optimized Amazon Linux AMI

以下是您可用来将最新的 Amazon ECS-optimized AMI 启动到集群中的多种方式：

- Amazon ECS 控制台首次运行向导使用最新的Amazon ECS-optimized AMI 启动容器实例。有关更多信息，请参阅 开始使用 Amazon ECS (p. 19)。
- 您可以按照启动 Amazon ECS 容器实例 (p. 165)中的过程在 Amazon EC2 控制台中手动启动容器实例。您也可以在下表之一中选择与您的集群区域对应的 EC2 Console Link (EC2 控制台链接)。
- 您可以使用 Systems Manager API 以编程方式检索当前的Amazon ECS-optimized AMI。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。
- 根据您选择的变体，使用 AWS CLI、AWS 开发工具包或 AWS CloudFormation 模板通过下面的当前 Amazon ECS-optimized AMI ID 启动您的实例。

Amazon Linux 2

当前 Amazon ECS-optimized Amazon Linux 2 AMI 包括：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux 2 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0dca97e7cde7be3d5	Launch instance
us-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0fac5486e4cff37f4	Launch instance
us-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0c6e63b58aac1043e	Launch instance
us-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0e5e051fd0b505db6	Launch instance
ap-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-02252d984c7e3595d	Launch instance
ap-northeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04a735b489d2a0320	Launch instance
ap-northeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0accbb5aa909be7b6	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-south-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a8bf4e187339e2c1	Launch instance
ap-southeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-05c6d22d98f9747fe	Launch instance
ap-southeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-039bb4c3a7946ce19	Launch instance
ca-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-01c07ee95e77abba8	Launch instance
eu-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0650e7d86452db33b	Launch instance
eu-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-059aa04f0c253adfb	Launch instance
eu-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0ae254c8a2d3346a7	Launch instance
eu-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0de1dc478496a9e2b	Launch instance
eu-west-3	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0d260f3e5ccd06043	Launch instance
sa-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-038707d64e5b8e7ba	Launch instance
us-gov-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a224902b35f8adfe	Launch instance
us-gov-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04c68165	Launch instance
cn-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance
cn-northwest-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance

Amazon Linux 2 (arm64)

当前 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 包括：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-02a59fb754c85ab16	Launch instance
us-east-1	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0b3b892651e52f0b1	Launch instance
us-west-2	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0860edcdc9c9533e2	Launch instance
eu-west-1	amzn2-ami-ecs-hvm-2.0.20190709-arm64-ebs	ami-0af65bdd9a59a3172	Launch instance

Amazon Linux 2 (GPU)

当前 Amazon ECS GPU-optimized AMI 包含以下内容：

- 最新的最简版本的 Amazon Linux 2
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)
- 建议的 NVIDIA 驱动程序版本 (`418.40.04`)
- NVIDIA 容器运行时挂钩版本 (`v1.4.0-1`)
- 建议的 CUDA 版本 (`9.2.88`)

下表按区域列出了当前的 Amazon ECS GPU-optimized AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0522cd69e8a331c3a	Launch instance
us-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0875a1e6c2db1cdc9	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
us-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0dfe31c7ed6577ebf	Launch instance
us-west-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0bf12263037af5756	Launch instance
ap-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0efcda0accbf5a0f4	Launch instance
ap-northeast-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-041843b72dde36df4	Launch instance
ap-northeast-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0e18b01ac7608adcd	Launch instance
ap-south-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0c5cdf93d467bec8a	Launch instance
ap-southeast-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-07b25dd477ea57f82	Launch instance
ap-southeast-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-07f990e4e7551b774	Launch instance
ca-central-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-08e1aab6a2cfd87ad	Launch instance
eu-central-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-096ccf1edd62587ce	Launch instance
eu-north-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0576489203c1c4cce	Launch instance
eu-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-03867df65b0e5abb2	Launch instance
eu-west-2	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0faecf24ba37edfce	Launch instance
eu-west-3	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0dec5973b167d5ff2	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
sa-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0a39851053709497a	Launch instance
us-gov-east-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0365a6c905c4ba391	Launch instance
us-gov-west-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-f8b8ff99	Launch instance
cn-north-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-01143a340285bc1fb	Launch instance
cn-northwest-1	amzn2-ami-ecs-gpu-hvm-2.0.20190709-x86_64-ebs	ami-0240ef2b16197fef2	Launch instance

Amazon Linux AMI

当前 Amazon ECS-optimized Amazon Linux AMI 包括：

- 最新的最简版本的 Amazon Linux AMI
- 最新版本的 Amazon ECS 容器代理 (`1.29.1`)
- 最新 Amazon ECS 容器代理的 Docker 的推荐版本 (`18.06.1-ce`)
- 用于运行和监控 Amazon ECS 代理的 `ecs-init` 程序包的最新版本 (`1.29.1-1`)

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0151b45908571e14e	Launch instance
us-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0c09d65d2051adab2	Launch instance
us-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0b4f28359911d58b6	Launch instance
us-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-077368b501184adb9	Launch instance
ap-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0eeaf74d330e7e8ca	Launch instance
ap-northeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-084d4626fec6f3cb2	Launch instance
ap-northeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-03c5d1f1ba6ec1b7d	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-south-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-09f6bc049f323302	Launch instance
ap-southeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-01813416d767d88b	Launch instance
ap-southeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-07fcafbb9f3a4f7b	Launch instance
ca-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-011742239f8f42f4	Launch instance
eu-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0bceb1887b6b3713	Launch instance
eu-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0d7012b0cae33c04	Launch instance
eu-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-02b9d5ef54d57fb7	Launch instance
eu-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0f49b2a9014635032	Launch instance
eu-west-3	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0db8d6e7dcf3cb362	Launch instance
sa-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0ad0be326813ddab6	Launch instance
us-gov-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-033b28d8ef5fe3f72	Launch instance
us-gov-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-b0bafdd1	Launch instance
cn-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-041f913b515304af9	Launch instance
cn-northwest-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0be42f3b41b17b512	Launch instance

Windows Server 2019

当前 Amazon ECS-optimized Windows 2019 AMI 包括：

- 最新版本的 Microsoft Windows Server 2019
- Docker EE 版本 `18.09.8`
- Amazon ECS 容器代理版本 `1.29.0`

下表按区域列出了当前的 Amazon ECS-optimized Windows 2019 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-05c650b3e1759293	Launch instance
us-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-00f3f16f1d05e53a	Launch instance
us-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-08f56c127cc9ae9c4	Launch instance
us-west-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0c37f25e59d752c61	Launch instance
ap-northeast-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0771bfb73bc362235	Launch instance
ap-northeast-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0849ba95174008d74	Launch instance
ap-south-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0dfe1bb9ed2dd2ff1	Launch instance
ap-southeast-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-06daa78e84b80091a	Launch instance
ap-southeast-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-071e43d09ba6f2479	Launch instance
ca-central-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0ed093f8cd0bb6f54	Launch instance
eu-central-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-020bded18f4df2c7d	Launch instance
eu-north-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0269f4d7fa129ddcb	Launch instance
eu-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e3604f46f9ccc506	Launch instance
eu-west-2	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0746c037f8889f6e8	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
eu-west-3	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-022ae3cb06a066371	Launch instance
sa-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-094ff1452e5849bf8	Launch instance
us-gov-east-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e238cac0ef640949	Launch instance
us-gov-west-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-13fcbd72	Launch instance
cn-north-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-02728cc741b470320	Launch instance
cn-northwest-1	Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	ami-0e1cafc0e9ae3fafa	Launch instance

Windows Server 2016

当前 Amazon ECS-optimized Windows 2016 AMI 包括：

- 最新版本的 Microsoft Windows Server 2016
- Docker EE 版本 `18.09.8`
- Amazon ECS 容器代理版本 `1.29.0`

下表按区域列出了当前的 Amazon ECS-optimized Windows 2016 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0ca3903548e4a8631	Launch instance
us-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0831d21a034cb33e4	Launch instance
us-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-08b04db86cc171254	Launch instance
us-west-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0b380fdc264da2fc2	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-northeast-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0d3f659d5bab5c8e5	Launch instance
ap-northeast-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0d4fe34a4a5cb98ba	Launch instance
ap-south-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0add8558b7afe4950	Launch instance
ap-southeast-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-001b756da3bc277be	Launch instance
ap-southeast-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-063f16052f2b588a2	Launch instance
ca-central-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-07cc1c48c86d988ca	Launch instance
eu-central-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0644eb70b836aa5ba	Launch instance
eu-north-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-038a1a6d22e486228	Launch instance
eu-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0e41be650397b1316	Launch instance
eu-west-2	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0884e795ec6511040	Launch instance
eu-west-3	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-03f921d23a15ff701	Launch instance
sa-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-06955dc02fc2a3b7b	Launch instance
us-gov-east-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-04c0f4fe213678f3b	Launch instance
us-gov-west-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-38fdbc59	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
cn-north-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-010da5256208bcb66	Launch instance
cn-northwest-1	Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	ami-0c1033d4d4607074 7	Launch instance

有关早期版本和对应的 Docker 和 Amazon ECS 容器代理版本的更多信息，请参阅Amazon ECS-Optimized Amazon Linux AMI 容器代理版本 (p. 208)。

经 Amazon ECS 优化的 AMI 版本

此主题列出了 Amazon ECS-optimized AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理、Docker 和 `ecs-init` 程序包的相应版本。

每个变体的 Amazon ECS-optimized AMI 元数据（包括 AMI ID）均可通过编程方式检索。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。

主题

- Amazon ECS-Optimized Amazon Linux 2 AMI 版本 (p. 146)
- Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 版本 (p. 147)
- Amazon ECS GPU-optimized AMI 版本 (p. 148)
- Amazon ECS-optimized Amazon Linux AMI 版本 (p. 148)
- Amazon ECS-optimized Windows 2019 AMI 版本 (p. 151)
- Amazon ECS-optimized Windows 2016 AMI 版本 (p. 151)

Amazon ECS-Optimized Amazon Linux 2 AMI 版本

下表列出了Amazon ECS-optimized Amazon Linux 2 AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理、Docker 和 `ecs-init` 程序包的相应版本。

Amazon ECS-optimized Amazon Linux 2 AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
20190709	1.29.1	18.06.1-ce	1.29.1-1
20190614	1.29.0	18.06.1-ce	1.29.0-1
20190607	1.29.0	18.06.1-ce	1.29.0-1
20190603	1.28.1	18.06.1-ce	1.28.1-2
20190510	1.28.0	18.06.1-ce	1.28.0-1
20190402	1.27.0	18.06.1-ce	1.27.0-1
20190301	1.26.0	18.06.1-ce	1.26.0-1
20190215	1.25.3	18.06.1-ce	1.25.3-1
20190204	1.25.2	18.06.1-ce	1.25.2-1

Amazon ECS-optimized Amazon Linux 2 AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
20190127	1.25.1	18.06.1-ce	1.25.1-1
20190118	1.25.0	18.06.1-ce	1.25.0-1
20190107	1.24.0	18.06.1-ce	1.24.0-1
20181112	1.22.0	18.06.1-ce	1.22.0-1
20181016	1.20.3	18.06.1-ce	1.21.0-1

可将 AWS CLI 与以下命令结合使用来检索当前 Amazon ECS-optimized Amazon Linux 2 AMI ：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended
```

Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 版本

下表列出了Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理、Docker 和 `ecs-init` 程序包的相应版本。

Amazon ECS-optimized Amazon Linux 2 (arm64) AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
20190709	1.29.1	18.06.1-ce	1.29.1-1
20190617	1.29.0	18.06.1-ce	1.29.0-1
20190607	1.29.0	18.06.1-ce	1.29.0-1
20190603	1.28.1	18.06.1-ce	1.28.1-2
20190510	1.28.0	18.06.1-ce	1.28.0-1
20190403	1.27.0	18.06.1-ce	1.27.0-1
20190301	1.26.0	18.06.1-ce	1.26.0-1
20190215	1.25.3	18.06.1-ce	1.25.3-1
20190204	1.25.2	18.06.1-ce	1.25.2-1
20190127	1.25.1	18.06.1-ce	1.25.1-1
20190119	1.25.0	18.06.1-ce	1.25.0-1
20181120	1.22.0	18.06.1-ce	1.22.0-1

可将 AWS CLI 与以下命令结合使用来检索当前Amazon ECS-optimized Amazon Linux 2 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/arm64/
recommended
```

Amazon ECS GPU-optimized AMI 版本

下表列出了Amazon ECS GPU-optimized AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理、Docker、 `ecs-init` 程序包和 NVIDIA 驱动程序的相应版本。

Amazon ECS GPU-optimized AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version	NVIDIA driver version
20190709	`1.29.1`	`18.06.1-ce`	`1.29.1-1`	`418.40.04`
20190614	`1.29.0`	`18.06.1-ce`	`1.29.0-1`	`418.40.04`
20190607	`1.29.0`	`18.06.1-ce`	`1.29.0-1`	`418.40.04`
20190603	`1.28.1`	`18.06.1-ce`	`1.28.1-2`	`418.40.04`
20190510	`1.28.0`	`18.06.1-ce`	`1.28.0-1`	`418.40.04`
20190402	`1.27.0`	`18.06.1-ce`	`1.27.0-1`	`418.40.04`
20190321	`1.26.0`	`18.06.1-ce`	`1.26.0-1`	`410.104`
20190301	`1.26.0`	`18.06.1-ce`	`1.26.0-1`	`396.26`
20190215	`1.25.3`	`18.06.1-ce`	`1.25.3-1`	`396.26`
20190204	`1.25.2`	`18.06.1-ce`	`1.25.2-1`	`396.26`
20190127	`1.25.1`	`18.06.1-ce`	`1.25.1-1`	`396.26`
20190118	`1.25.0`	`18.06.1-ce`	`1.25.0-1`	`396.26`

您可以将 AWS CLI 与以下命令结合使用来检索当前 Amazon ECS GPU-optimized AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended
```

Amazon ECS-optimized Amazon Linux AMI 版本

下表列出了 Amazon ECS-optimized Amazon Linux AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理、Docker 和 `ecs-init` 程序包的相应版本。

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
2018.03.v	`1.29.1`	`18.06.1-ce`	`1.29.1-1`
2018.03.u	`1.29.0`	`18.06.1-ce`	`1.29.0-1`
2018.03.t	`1.29.0`	`18.06.1-ce`	`1.29.0-1`
2018.03.s	`1.28.1`	`18.06.1-ce`	`1.28.1-2`
2018.03.q	`1.28.0`	`18.06.1-ce`	`1.28.0-1`
2018.03.p	`1.27.0`	`18.06.1-ce`	`1.27.0-1`

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
2018.03.o	`1.26.0`	`18.06.1-ce`	`1.26.0-1`
2018.03.n	`1.25.3`	`18.06.1-ce`	`1.25.3-1`
2018.03.m	`1.25.2`	`18.06.1-ce`	`1.25.2-1`
2018.03.l	`1.25.1`	`18.06.1-ce`	`1.25.1-1`
2018.03.k	`1.25.0`	`18.06.1-ce`	`1.25.0-1`
2018.03.j	`1.24.0`	`18.06.1-ce`	`1.24.0-1`
2018.03.i	`1.22.0`	`18.06.1-ce`	`1.22.0-1`
2018.03.h	`1.21.0`	`18.06.1-ce`	`1.21.0-1`
2018.03.g	`1.20.3`	`18.06.1-ce`	`1.20.3-1`
2018.03.f	`1.20.2`	`18.06.1-ce`	`1.20.2-1`
2018.03.e	`1.20.1`	`18.03.1-ce`	`1.20.1-1`
2018.03.d	`1.20.0`	`18.03.1-ce`	`1.20.0-1`
2018.03.c	`1.19.1`	`18.03.1-ce`	`1.19.1-1`
2018.03.b	`1.19.0`	`18.03.1-ce`	`1.19.0-1`
2018.03.a	`1.18.0`	`17.12.1-ce`	`1.18.0-1`
2017.09.l	`1.17.3`	`17.12.1-ce`	`1.17.3-1`
2017.09.k	`1.17.2`	`17.12.0-ce`	`1.17.2-1`
2017.09.j	`1.17.2`	`17.12.0-ce`	`1.17.2-1`
2017.09.i	`1.17.1`	`17.09.1-ce`	`1.17.1-1`
2017.09.h	`1.17.0`	`17.09.1-ce`	`1.17.0-2`
2017.09.g	`1.16.2`	`17.09.1-ce`	`1.16.2-1`
2017.09.f	`1.16.1`	`17.06.2-ce`	`1.16.1-1`
2017.09.e	`1.16.1`	`17.06.2-ce`	`1.16.1-1`
2017.09.d	`1.16.0`	`17.06.2-ce`	`1.16.0-1`
2017.09.c	`1.15.2`	`17.06.2-ce`	`1.15.1-1`
2017.09.b	`1.15.1`	`17.06.2-ce`	`1.15.1-1`
2017.09.a	`1.15.0`	`17.06.2-ce`	`1.15.0-4`
2017.03.g	`1.14.5`	`17.03.2-ce`	`1.14.5-1`
2017.03.f	`1.14.4`	`17.03.2-ce`	`1.14.4-1`
2017.03.e	`1.14.3`	`17.03.1-ce`	`1.14.3-1`

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
2017.03.d	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2017.03.c	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2017.03.b	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2016.09.g	`1.14.1`	`1.12.6`	`1.14.1-1`
2016.09.f	`1.14.0`	`1.12.6`	`1.14.0-2`
2016.09.e	`1.14.0`	`1.12.6`	`1.14.0-1`
2016.09.d	`1.13.1`	`1.12.6`	`1.13.1-2`
2016.09.c	`1.13.1`	`1.11.2`	`1.13.1-1`
2016.09.b	`1.13.1`	`1.11.2`	`1.13.1-1`
2016.09.a	`1.13.0`	`1.11.2`	`1.13.0-1`
2016.03.j	`1.13.0`	`1.11.2`	`1.13.0-1`
2016.03.i	`1.12.2`	`1.11.2`	`1.12.2-1`
2016.03.h	`1.12.1`	`1.11.2`	`1.12.1-1`
2016.03.g	`1.12.0`	`1.11.2`	`1.12.0-1`
2016.03.f	`1.11.1`	`1.11.2`	`1.11.1-1`
2016.03.e	`1.11.0`	`1.11.2`	`1.11.0-1`
2016.03.d	`1.10.0`	`1.11.1`	`1.10.0-1`
2016.03.c	`1.10.0`	`1.11.1`	`1.10.0-1`
2016.03.b	`1.9.0`	`1.9.1`	`1.9.0-1`
2016.03.a	`1.8.2`	`1.9.1`	`1.8.2-1`
2015.09.g	`1.8.1`	`1.9.1`	`1.8.1-1`
2015.09.f	`1.8.0`	`1.9.1`	`1.8.0-1`
2015.09.e	`1.7.1`	`1.9.1`	`1.7.1-1`
2015.09.d	`1.7.1`	`1.9.1`	`1.7.1-1`
2015.09.c	`1.7.0`	`1.7.1`	`1.7.0-1`
2015.09.b	`1.6.0`	`1.7.1`	`1.6.0-1`
2015.09.a	`1.5.0`	`1.7.1`	`1.5.0-1`
2015.03.g	`1.4.0`	`1.7.1`	`1.4.0-2`
2015.03.f	`1.4.0`	`1.6.2`	`1.4.0-1`
2015.03.e	`1.3.1`	`1.6.2`	`1.3.1-1`

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
2015.03.d	1.2.1	1.6.2	1.2.0-2
2015.03.c	1.2.0	1.6.2	1.2.0-1
2015.03.b	1.1.0	1.6.0	1.0-3
2015.03.a	1.0.0	1.5.0	1.0-1

Amazon ECS-optimized Windows 2019 AMI 版本

下表列出了 Amazon ECS-optimized Windows 2019 AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理 和 Docker 的相应版本。

Amazon ECS-optimized Windows 2019 AMI	Amazon ECS container agent version	Docker version
2019.07.19	1.29.0	18.09.8
2019.05.10	1.27.0	18.09.4

可将 AWS CLI 与以下命令结合使用来检索当前 Amazon ECS-optimized Windows 2019 AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Amazon ECS-optimized Windows 2016 AMI 版本

下表列出了 Amazon ECS-optimized Windows 2016 AMI 的当前版本和早期版本，以及 Amazon ECS 容器代理和 Docker 的相应版本。

Amazon ECS-optimized Windows 2016 AMI	Amazon ECS container agent version	Docker version
2019.07.19	1.29.0	18.09.8
2019.03.07	1.26.0	18.03.1

可将 AWS CLI 与以下命令结合使用来检索当前 Amazon ECS-optimized Windows 2016 AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

AMI 存储配置

下面介绍了每个经 Amazon ECS 优化的 AMI 的存储配置。

主题

- Amazon Linux 2 存储配置 (p. 152)
- Amazon ECS-optimized Amazon Linux AMI 存储配置 (p. 152)

Amazon Linux 2 存储配置

默认情况下，基于 Amazon Linux 2 的经 Amazon ECS 优化的 AMI（Amazon ECS-optimized Amazon Linux 2 AMI、Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU-optimized AMI）附带 30 GiB 的根卷。您可以在启动时修改 30 GiB 的根卷大小，以增加您的容器实例上的可用存储。此存储用于操作系统和 Docker 映像与元数据。

Amazon ECS-optimized Amazon Linux 2 AMI 的默认文件系统是 `ext4`，Docker 使用 `overlay2` 存储驱动程序。有关更多信息，请参阅 Docker 文档中的使用 OverlayFS 存储驱动程序。

Amazon ECS-optimized Amazon Linux AMI 存储配置

默认情况下，Amazon ECS-optimized Amazon Linux AMI 附带 30GiB 的总存储量。您可以在启动时修改此值，以增加容器实例上的可用存储。此存储用于操作系统和 Docker 映像与元数据。下面几节介绍基于 AMI 版本的Amazon ECS-optimized Amazon Linux AMI 的存储配置。

版本 2015.09.d 及更高版本

`2015.09.d` 版及更高版本中的Amazon ECS-optimized Amazon Linux AMI 使用 8GiB 卷进行启动，该卷用于在 `/dev/xvda` 挂载并装载为文件系统根的操作系统。另外还有一个在 `/dev/xvdcz` 挂载并供 Docker 用于映像和元数据存储的 22 GiB 卷。此卷将配置为逻辑卷管理 (LVM) 设备，并且可由 Docker 通过 `devicemapper` 后端直接访问。因为未装载此卷，所以无法使用标准存储信息命令（如 df -h）确定可用存储。但您可以使用 LVM 命令和 docker info 通过以下过程查找可用存储。有关更多信息，请参阅 Linux 文档项目中的 LVM HOWTO。

> **Note**
>
> 您可以通过在启动实例时更改其块储存设备映射设置来增加这些默认卷大小；但是，您不能指定小于默认值的卷大小。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的块储存设备映射。

docker-storage-setup 实用程序在实例启动时为 Docker 配置 LVM 卷组和逻辑卷。默认情况下，docker-storage-setup 创建一个名为 `docker` 的卷组，并将 `/dev/xvdcz` 作为物理卷添加到该组。然后，它将创建一个名为 `docker-pool` 的逻辑卷，该卷使用卷组中 99% 的可用存储。剩余 1% 的可用存储将为元数据保留。

> **Note**
>
> 早期版本的Amazon ECS-optimized Amazon Linux AMI（`2015.09.d` 至 `2016.03.a`）将创建一个使用卷组中 40% 的可用存储的逻辑卷。当逻辑卷的容量已占用 60% 时，逻辑卷的大小将增加 20%。

确定 Docker 的可用存储

- 您可以使用 LVM 命令、vgs 和 lvs 或 docker info 命令来查看 Docker 的可用存储。

  > **Note**
  >
  > LVM 命令输出显示以 GiB（2^30 字节）为单位的存储值，而 docker info 显示以 GB（10^9 字节）为单位的存储值。

  a. 您可以使用 vgs 命令查看卷组中的可用存储。此命令显示卷组的总大小以及卷组中可用于增大逻辑卷的可用空间。以下示例显示一个 22 GiB 卷，其可用空间为 204 MiB。

  ```
  [ec2-user ~]$ sudo vgs
  ```

  输出：

  ```
    VG     #PV #LV #SN Attr   VSize  VFree
  ```

```
  docker   1   1   0 wz--n- 22.00g 204.00m
```

b. 您可以使用 lvs 命令查看逻辑卷中的可用空间。以下示例显示一个大小为 21.75 GiB 的逻辑卷，该卷的 7.63% 的空间已占用。此逻辑卷能够增大，直到卷组中不再有可用空间。

```
[ec2-user@ ~]$ sudo lvs
```

输出：

```
  LV          VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
 Convert
  docker-pool docker twi-aot--- 21.75g             7.63   4.96
```

c. docker info 命令还提供有关其占用的数据空间以及可用数据空间的信息。不过，其可用空间值基于其正在使用的逻辑卷的大小。

> **Note**
>
> 因为 docker info 以 GB（10^9 字节）而不是 GiB（2^30 字节）为单位显示存储值，所以，对于相同的存储量，此处显示的值看起来大于使用 lvs 命令显示的值。但是，这两个值是相等的 (23.35 GB = 21.75 GiB)。

```
[ec2-user ~]$ docker info | grep "Data Space"
```

输出：

```
 Data Space Used: 1.782 GB
 Data Space Total: 23.35 GB
 Data Space Available: 21.57 GB
```

扩展 Docker 逻辑卷

向您的容器实例添加存储的最简单方式是终止现有实例，然后启动具有更大的数据存储卷的新实例。但如果您无法执行此操作，则可以向 Docker 使用的卷组添加存储，然后执行这些步骤来扩展其逻辑卷。

> **Note**
>
> 如果您的容器实例存储是填满速度过快，则可采取以下几个措施来减小影响：
>
> - （Amazon ECS 容器代理 1.8.0 及更高版本）减少已停止或已退出的容器在您的容器实例上的保留时间。`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 代理配置变量可设置从任务停止到 Docker 容器被删除所需等待的时间段 (默认情况下，此值为 3 小时)。这将删除 Docker 容器数据。如果此值设置得过低，您可能无法检查已停止的容器或无法在日志删除前查看它们。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。
> - 从您的容器实例中删除未运行的容器和未使用的映像。您可以使用以下示例命令来手动删除已停止的容器和未使用的映像。稍后将无法检查已删除的容器，而且必须先重新拉取已删除的映像，然后再从中启动新容器。
>
> 要删除未运行的容器，请在容器实例上执行下面的命令：
>
> ```
> $ docker rm $(docker ps -aq)
> ```
>
> 要删除未使用的映像，请在容器实例上执行下面的命令：
>
> ```
> $ docker rmi $(docker images -q)
> ```

- 删除容器内未使用的数据块。您可以在任何正在运行的容器上使用下面的命令运行 fstrim 来丢弃该容器文件系统未使用的任何数据块。

```
$ sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' |
 xargs -IZ fstrim /proc/Z/root/"
```

1. 在容器实例所在的可用区中创建新的 Amazon EBS 卷。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的创建 Amazon EBS 卷。
2. 将卷附加到您的容器实例。Docker 数据卷的默认位置为 `/dev/xvdcz`。为了保持一致，从该设备名称 (例如，`/dev/xvdcy`) 以逆字母顺序挂载其他卷。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的将 Amazon EBS 卷挂载到实例。
3. 使用 SSH 连接到您的容器实例。有关更多信息，请参阅 连接到您的容器实例 (p. 180)。
4. 检查您的 `docker-pool` 逻辑卷的大小。以下示例显示一个大小为 409.19 GiB 的逻辑卷。

```
[ec2-user ~]$ sudo lvs
```

输出：

```
  LV          VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync
 Convert
  docker-pool docker twi-aot--- 409.19g             0.16   0.08
```

5. 检查您的卷组中的当前可用空间。以下示例在 `VFree` 列中显示 612.75 GiB。

```
[ec2-user ~]$ sudo vgs
```

输出：

```
  VG     #PV #LV #SN Attr   VSize    VFree
  docker   1   1   0 wz--n- 1024.00g 612.75g
```

6. 将新卷添加到 `docker` 卷组，并替换您将新卷添加到的设备名称。在此示例中，1 TiB 卷之前已添加并挂载到 `/dev/xvdcy`。

```
[ec2-user ~]$ sudo vgextend docker /dev/xvdcy
  Physical volume "/dev/sdcy" successfully created
  Volume group "docker" successfully extended
```

7. 使用 vgs 命令验证卷组大小是否已增加。`VFree` 列应显示增加的存储大小。以下示例现在在 `VFree` 列中有 1.6 TiB，此值比其之前的值大 1 TiB。您的 `VFree` 列应是原始 `VFree` 值和您附加的卷的大小的和。

```
[ec2-user ~]$ sudo vgs
```

输出：

```
  VG     #PV #LV #SN Attr   VSize VFree
  docker   2   1   0 wz--n- 2.00t 1.60t
```

8. 利用之前添加的卷的大小扩展 `docker-pool` 逻辑卷。以下命令向逻辑卷添加 1024 GiB，输入形式为 *1024G*。

```
[ec2-user ~]$ sudo lvextend -L+1024G /dev/docker/docker-pool
```

输出：

```
  Size of logical volume docker/docker-pool_tdata changed from 409.19 GiB (104752
 extents) to 1.40 TiB (366896 extents).
  Logical volume docker-pool successfully resized
```

9. 验证逻辑卷的大小是否已增加。

```
[ec2-user ~]$ sudo lvs
```

输出：

```
  LV          VG     Attr       LSize Pool Origin Data%  Meta%  Move Log Cpy%Sync
 Convert
  docker-pool docker twi-aot--- 1.40t             0.04   0.12
```

10. （可选）验证 docker info 是否也识别已添加的存储空间。

    Note

    因为 docker info 以 GB（10^9 字节）而不是 GiB（2^30 字节）为单位显示存储值，所以，对于相同的存储量，此处显示的值看起来大于使用 lvs 命令显示的值。但是，这两个值是相等的 (1.539 TB =1.40 TiB)。

```
[ec2-user ~]$ docker info | grep "Data Space"
```

输出：

```
 Data Space Used: 109.6 MB
 Data Space Total: 1.539 TB
 Data Space Available: 1.539 TB
```

2015.09.c 版及早期版本

`2015.09.c` 版及更早版本中的Amazon ECS-optimized Amazon Linux AMI 使用单个 30GiB 卷进行启动，该卷在 `/dev/xvda` 挂载并安装为文件系统根。此卷共享操作系统和所有 Docker 映像和元数据。您可以使用标准存储信息命令（如 df -h）确定容器实例上的可用存储。

在不停止通过这些 AMI 启动的实例的情况下，没有可行的方法向这些实例添加存储（Docker 可使用）。如果发现容器实例需要的存储量大于默认值 30 GiB，应终止每个实例。然后，在其位置上使用最新的Amazon ECS-optimized Amazon Linux AMI 和足够大的数据存储卷启动另一个实例。

检索 Amazon ECS-Optimized AMI 元数据

不同Amazon ECS-optimized AMI 的 AMI ID、映像名称、操作系统、容器代理版本和运行时版本可通过查询 Systems Manager Parameter Store API 以编程方式检索。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameters 和 GetParametersByPath。

Note

用户账户必须具有以下 IAM 权限才能检索Amazon ECS-optimized AMI 元数据。这些权限已添加到 `AmazonECS_FullAccess` IAM 策略。

- ssm:GetParameters

- ssm:GetParameter
- ssm:GetParametersByPath

以下是参数名称的格式。

- Amazon ECS-optimized Amazon Linux 2 AMI 元数据：

```
<version>
```

- Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 元数据：

```
<version>
```

- Amazon ECS GPU-optimized AMI 元数据：

```
<version>
```

- Amazon ECS-optimized Amazon Linux AMI 元数据：

```
<version>
```

- Amazon ECS-optimized Windows 2019 AMI 元数据：

```
windows_server/2019/english/full/<version>
```

- Amazon ECS-optimized Windows 2016 AMI 元数据：

```
windows_server/2016/english/full/<version>
```

以下参数名称格式通过使用 `recommended` 检索最新稳定的Amazon ECS-optimized Amazon Linux 2 AMI 的元数据。

```
recommended
```

以下是针对参数值返回的 JSON 对象的示例。

```
{
 "schema_version": 1,
 "image_name": "amzn2-ami-ecs-hvm-2.0.20181017-x86_64-ebs",
 "image_id": "ami-04a4fb062c609f55b",
 "os": "Amazon Linux 2",
 "ecs_runtime_version": "Docker version 18.06.1-ce",
 "ecs_agent_version": "1.21.0"
}
```

上述输出中的每个字段都可作为子参数查询。通过将子参数名称追加到所选 AMI 的路径来构造子参数的参数路径。可用子参数如下：

- `schema_version`
- `image_id`
- `image_name`
- `os`
- `ecs_agent_version`
- `ecs_runtime_version`

以下参数名称格式通过使用子参数 `image_id` 检索最新稳定的Amazon ECS-optimized Amazon Linux 2 AMI 的映像 ID。

```
recommended/image_id
```

以下参数名称格式通过指定 AMI 名称来检索特定Amazon ECS-optimized AMI 版本的元数据。

- Amazon ECS-optimized Amazon Linux 2 AMI 元数据：

```
amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

- Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 元数据：

```
amzn2-ami-ecs-hvm-2.0.20181120-arm64-ebs
```

- Amazon ECS-optimized Amazon Linux AMI 元数据：

```
amzn-ami-2017.09.l-amazon-ecs-optimized
```

- Amazon ECS-optimized Windows 2019 AMI 元数据：

```
/aws/service/ecs/optimized-ami/windows_server/2019/english/full/2019.05.10
```

- Amazon ECS-optimized Windows 2016 AMI 元数据：

```
/aws/service/ecs/optimized-ami/windows_server/2016/english/full/2019.03.07
```

> **Note**
>
> 所有版本的Amazon ECS-optimized Amazon Linux 2 AMI 都可供检索。只能检索经 Amazon ECS 优化的 AMI 版本 `amzn-ami-2017.09.l-amazon-ecs-optimized` (Linux) 和 `Windows_Server-2016-English-Full-ECS_Optimized-2018.03.26` (Windows) 及更高版本。有关更多信息，请参阅 经 Amazon ECS 优化的 AMI 版本 (p. 146)。

Example 检索最新稳定的Amazon ECS-optimized AMI 的元数据

您可将 AWS CLI 与以下 AWS CLI 命令结合使用来检索最新稳定的Amazon ECS-optimized AMI。

- 对于Amazon ECS-optimized Amazon Linux 2 AMI：

```
aws ssm get-parameters --names recommended --region us-east-1
```

输出：

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-
hvm-2.0.20181017-x86_64-ebs\",\"image_id\":\"ami-0a6be20ed8ce1f055\",\"os\":\"Amazon
 Linux 2\",\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version\":
\"1.21.0\"}",
            "Version": 1,
            "LastModifiedDate": 1539908415.817,
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/recommended"
```

```
        }
    ],
    "InvalidParameters": []
}
```

- 对于Amazon ECS-optimized Amazon Linux 2 (arm64) AMI：

```
aws ssm get-parameters --names recommended --region us-east-1
```

输出：

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/arm64/recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-
hvm-2.0.20181120-arm64-ebs\",\"image_id\":\"ami-053b2a8c2f3e87928\",\"os\":\"Amazon
 Linux 2\",\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version\":
\"1.22.0\"}",
            "Version": 1,
            "LastModifiedDate": 1542745522.454,
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/arm64/recommended"
        }
    ],
    "InvalidParameters": []
}
```

- 对于Amazon ECS GPU-optimized AMI：

```
aws ssm get-parameters --names recommended --region us-east-1
```

输出：

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-gpu-
hvm-2.0.20190118-x86_64-ebs\",\"image_id\":\"ami-0f8776282f835efc5\",\"os\":\"Amazon
 Linux 2\",\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version\":
\"1.25.0\"}",
            "Version": 1,
            "LastModifiedDate": 1548369045.401,
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended"
        }
    ],
    "InvalidParameters": []
}
```

- 对于Amazon ECS-optimized Amazon Linux AMI：

```
aws ssm get-parameters --names recommended --region us-east-1
```

输出：

```
{
```

```
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux/recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"amzn-ami-2018.03.h-
amazon-ecs-optimized\",\"image_id\":\"ami-07eb698ce660402d2\",\"os\":\"Amazon Linux\",
\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version\":\"1.21.0\"}",
            "Version": 11,
            "LastModifiedDate": 1539892113.403,
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ecs/optimized-ami/
amazon-linux/recommended"
        }
    ],
    "InvalidParameters": []
}
```

- 对于 Amazon ECS-optimized Windows 2019 AMI：

```
aws ssm get-parameters --names recommended --region us-east-1
```

输出:

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/windows_server/2019/english/full/
recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"Windows_Server-2019-
English-Full-ECS_Optimized-2019.05.10\",\"image_id\":\"ami-0a4548e9bef884a63\",\"os\":
\"Windows_Server-2019-English-Full\",\"ecs_runtime_version\":\"Docker version 18.09.4\",
\"ecs_agent_version\":\"1.27.0\"}",
            "Version": 1,
            "LastModifiedDate": 1557785029.622,
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
windows_server/2019/english/full/recommended"
        }
    ],
    "InvalidParameters": []
}
```

- 对于 Amazon ECS-optimized Windows 2016 AMI：

```
aws ssm get-parameters --names recommended --re
```

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/windows_server/2016/english/full/
recommended",
            "Type": "String",
            "Value": "{\"schema_version\":1,\"image_name\":\"Windows_Server-2016-
English-Full-ECS_Optimized-2019.03.07\",\"image_id\":\"ami-00c56e74f090d6f65\",\"os\":
\"Windows_Server-2016-English-Full\",\"ecs_runtime_version\":\"Docker version 18.03.1-
ee-7\",\"ecs_agent_version\":\"1.26.0\"}",
            "Version": 12,
            "LastModifiedDate": 1552024433.093,
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
windows_server/2016/english/full/recommended"
        }
    ],
    "InvalidParameters": []
```

```
}
```

- 对于 Amazon ECS-optimized Windows 2019 AMI：使用 AWS PowerShell

```
Get-SSMParameter -Name recommended/image_id -region us-east-1
```

输出：

```
Name                                                                                     Type
   Value         Version
----                                                                                     ----
   -----         -------
/aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id
 String ami-0a4548e9 3
```

Example 检索特定Amazon ECS-optimized Amazon Linux AMI 版本的元数据

将 AWS CLI 与以下 AWS CLI 命令结合使用来检索特定Amazon ECS-optimized Amazon Linux AMI 版本的元数据。将 AMI 名称替换为要检索的Amazon ECS-optimized Amazon Linux AMI 的名称。有关可用版本的更多信息，请参阅 经 Amazon ECS 优化的 AMI 版本 (p. 146)。

```
aws ssm get-parameters --names amzn-ami-2017.09.l-amazon-ecs-optimized --region us-east-1
```

Example 使用 Systems Manager GetParametersByPath API 检索Amazon ECS-optimized Amazon Linux AMI 元数据

将 AWS CLI 与以下命令结合使用来通过 Systems Manager GetParametersByPath API 检索Amazon ECS-optimized Amazon Linux AMI 元数据。

```
aws ssm get-parameters-by-path --path  --region us-east-1
```

Example 检索最新推荐的Amazon ECS-optimized Amazon Linux AMI 的映像 ID

您可以通过使用子参数 `image_id` 检索最新推荐的Amazon ECS-optimized Amazon Linux AMI ID 的映像 ID。

```
aws ssm get-parameters --names recommended/image_id --region us-east-1
```

输出：


```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux/recommended/image_id",
            "Type": "String",
            "Value": "ami-07eb698ce660402d2",
            "Version": 10,
            "LastModifiedDate": 1539892113.519,
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ecs/optimized-ami/amazon-
linux/recommended/image_id"
        }
    ],
    "InvalidParameters": []
}
```


要仅检索 `image_id` 值，您可以查询特定参数值；例如：

```
aws ssm get-parameters --names recommended/image_id --region us-east-1 --query
 "Parameters[0].Value"
```

输出：

```
"ami-f9ac2f86"
```

Example 在 AWS CloudFormation 模板中使用最新推荐的Amazon ECS-optimized AMI

您可以通过引用 Systems Manager Parameter Store 名称在 AWS CloudFormation 模板中检索最新推荐的 Amazon ECS-optimized AMI；例如：

Amazon Linux 2：

```
Parameters:
  ECSAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: recommended/image_id
```

Windows：

```
Parameters:
  ECSAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: recommended/image_id
```

订阅Amazon ECS-Optimized Amazon Linux AMI 更新通知

Amazon ECS-optimized Amazon Linux AMI 接收代理更改的定期更新、Docker 版本更新和 Linux 内核安全更新。您可以订阅 AMI 更新 Amazon SNS 主题，以在新的Amazon ECS-optimized Amazon Linux AMI 可用时接收通知。通知以 Amazon SNS 支持的所有格式提供。

> **Note**
>
> 用户账户必须拥有 `sns::subscribe` IAM 权限才能订阅 SNS 主题。

您可以为 Amazon SQS 队列订阅此通知主题，但您必须使用位于同一区域内的主题 ARN。有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南 中的教程：为 Amazon SQS 队列订阅 Amazon SNS 主题。

您也可以使用 AWS Lambda 函数在收到通知时触发事件。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南 中的使用 Amazon SNS 通知调用 Lambda 函数。

每个区域的 Amazon SNS 主题 ARN 如下所示。

AWS 地区	Amazon SNS 主题 ARN
us-east-1	arn:aws:sns:us-east-1:177427601217:ecs-optimized-amazon-ami-update

AWS 地区	Amazon SNS 主题 ARN
us-east-2	arn:aws:sns:us-east-2:177427601217:ecs-optimized-amazon-ami-update
us-west-1	arn:aws:sns:us-west-1:177427601217:ecs-optimized-amazon-ami-update
us-west-2	arn:aws:sns:us-west-2:177427601217:ecs-optimized-amazon-ami-update
ap-east-1	arn:aws:sns:ap-east-1:177427601217:ecs-optimized-amazon-ami-update
ap-northeast-1	arn:aws:sns:ap-northeast-1:177427601217:ecs-optimized-amazon-ami-update
ap-northeast-2	arn:aws:sns:ap-northeast-2:177427601217:ecs-optimized-amazon-ami-update
ap-southeast-1	arn:aws:sns:ap-southeast-1:177427601217:ecs-optimized-amazon-ami-update
ap-southeast-2	arn:aws:sns:ap-southeast-2:177427601217:ecs-optimized-amazon-ami-update
ap-south-1	arn:aws:sns:ap-south-1:177427601217:ecs-optimized-amazon-ami-update
ca-central-1	arn:aws:sns:ca-central-1:177427601217:ecs-optimized-amazon-ami-update
eu-west-1	arn:aws:sns:eu-west-1:177427601217:ecs-optimized-amazon-ami-update
eu-west-2	arn:aws:sns:eu-west-2:177427601217:ecs-optimized-amazon-ami-update
eu-west-3	arn:aws:sns:eu-west-3:177427601217:ecs-optimized-amazon-ami-update
eu-central-1	arn:aws:sns:eu-central-1:177427601217:ecs-optimized-amazon-ami-update

AWS 地区	Amazon SNS 主题 ARN
sa-east-1	arn:aws:sns:sa-east-1:177427601217:ecs-optimized-amazon-ami-update

在 AWS 管理控制台中订阅 AMI 更新通知电子邮件

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 在区域列表中，选择与要订阅的主题 ARN 相同的区域。此示例使用 `us-west-2` 区域。
3. 在左侧导航窗格中，依次选择订阅和创建订阅。
4. 在创建订阅对话框中，对于主题 ARN，粘贴Amazon ECS-optimized Amazon Linux AMI 更新主题 ARN：`arn:aws:sns:us-west-2:177427601217:ecs-optimized-amazon-ami-update`。
5. 对于 Protocol，选择 Email。对于终端节点，请键入您可用于接收通知的电子邮件地址。
6. 选择 Create subscription。
7. 在您的电子邮件应用程序中，打开来自 AWS 通知的消息并打开链接以确认订阅。

   您的 Web 浏览器将显示来自 Amazon SNS 的确认响应。

使用 AWS CLI 订阅 AMI 更新通知电子邮件

1. 使用 AWS CLI 运行以下命令：

```
aws sns --region us-west-2 subscribe --topic-arn arn:aws:sns:us-
west-2:177427601217:ecs-optimized-amazon-ami-update --protocol email --notification-
endpoint your_email@your_domain.com
```

2. 在您的电子邮件应用程序中，打开来自 AWS 通知的消息并打开链接以确认订阅。

   您的 Web 浏览器将显示来自 Amazon SNS 的确认响应。

Amazon SNS 消息格式

示例 AMI 更新通知消息如下所示：


```
{
  "Type" : "Notification",
  "MessageId" : "e2534930-337d-5561-8636-1a2be5ba802e",
  "TopicArn" : "arn:aws:sns:us-west-2:917786371007:ecs-optimized-amazon-ami-update",
  "Message" : "{\"ECSAgent\":{\"ReleaseVersion\":\"1.17.2\"},\"ECSAmis\":[{\"ReleaseVersion
\":\"2017.09.j\",\"AgentVersion\":\"1.17.2\",\"ReleaseNotes\":\"This AMI includes the
 latest ECS agent 1.17.2\",\"OsType\":\"linux\",\"OperatingSystemName\":\"Amazon Linux
\",\"Regions\":{\"ap-northeast-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",
\"ImageId\":\"ami-bb5f13dd\"},\"ap-northeast-2\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-
optimized\",\"ImageId\":\"ami-3b19b455\"},\"ap-south-1\":{\"Name\":\"amzn-ami-2017.09.j-
amazon-ecs-optimized\",\"ImageId\":\"ami-9e91cff1\"},\"ap-southeast-1\":{\"Name\":\"amzn-
ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-f88ade84\"},\"ap-southeast-2\":
{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-a677b6c4\"},
\"ca-central-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-
db48cfbf\"},\"cn-north-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId
\":\"ami-ca508ca7\"},\"eu-central-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized
\",\"ImageId\":\"ami-3b7d1354\"},\"eu-west-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-
optimized\",\"ImageId\":\"ami-64c4871d\"},\"eu-west-2\":{\"Name\":\"amzn-ami-2017.09.j-
amazon-ecs-optimized\",\"ImageId\":\"ami-25f51242\"},\"eu-west-3\":{\"Name\":\"amzn-
ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-0356e07e\"},\"sa-east-1\":{\"Name
```

```
\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-da2c66b6\"},\"us-east-1\":
{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-cad827b7\"},\"us-
east-2\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":\"ami-ef64528a
\"},\"us-gov-west-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",\"ImageId\":
\"ami-cc3cb7ad\"},\"us-west-1\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-optimized\",
\"ImageId\":\"ami-29b8b249\"},\"us-west-2\":{\"Name\":\"amzn-ami-2017.09.j-amazon-ecs-
optimized\",\"ImageId\":\"ami-baa236c2\"}}}]}",
  "Timestamp" : "2018-03-09T00:25:43.483Z",
  "SignatureVersion" : "1",
  "Signature" : "XWox8GDGLRiCgDOXlo/
fG9Lu/88P8S0FL6M6oQYOmUFzkucuhoblsdea3BjqdCHcWR7qdhMPQnLpN7y9iBrWVUqdAGJrukAI8athvAS+4AQD/
V/QjrhsEnlj+GaiW
+ozAu006X6GopOzFGnCtPMROjCMrMonjz7Hpv/8KRuMZR3pyQYm5d4wWB7xBPYhUMuLoZ1V8YFs55FMtgQV/
YLhSYuEu0BP1GMtLQauxDkscOtPP/vjhGQLFx1Q9LTadcQiRHtNIBxWL87PSI+BVvkin6AL7PhksvdQ7FAgHfXsit
+6p8GyOvKCqaeBG7HZhR1AbpyVka7JSNRO/6ssyrlj1g==",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-433026a4050d206028891664da859041.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:917786371007:ecs-optimized-amazon-
ami-update:8ad8798e-3bbf-4490-89b1-76136fca61dc"
}
```


解析的 `Message` 值 (删除了转义引号) 如下所示：


```
{
  "ECSAgent": {
    "ReleaseVersion": "1.17.2"
  },
  "ECSAmis": [
    {
      "ReleaseVersion": "2017.09.j",
      "AgentVersion": "1.17.2",
      "ReleaseNotes": "This AMI includes the latest ECS agent 1.17.2",
      "OsType": "linux",
      "OperatingSystemName": "Amazon Linux",
      "Regions": {
        "ap-northeast-1": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-bb5f13dd"
        },
        "ap-northeast-2": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-3b19b455"
        },
        "ap-south-1": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-9e91cff1"
        },
        "ap-southeast-1": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-f88ade84"
        },
        "ap-southeast-2": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-a677b6c4"
        },
        "ca-central-1": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-db48cfbf"
        },
        "cn-north-1": {
          "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
          "ImageId": "ami-ca508ca7"
        },
        "eu-central-1": {
```

```
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-3b7d1354"
          },
          "eu-west-1": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-64c4871d"
          },
          "eu-west-2": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-25f51242"
          },
          "eu-west-3": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-0356e07e"
          },
          "sa-east-1": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-da2c66b6"
          },
          "us-east-1": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-cad827b7"
          },
          "us-east-2": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-ef64528a"
          },
          "us-gov-west-1": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-cc3cb7ad"
          },
          "us-west-1": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-29b8b249"
          },
          "us-west-2": {
            "Name": "amzn-ami-2017.09.j-amazon-ecs-optimized",
            "ImageId": "ami-baa236c2"
          }
        }
      }
    ]
}
```

启动 Amazon ECS 容器实例

您可以使用 AWS 管理控制台启动 Amazon ECS 容器实例，如本主题中所述。开始之前，请确保您已完成Amazon ECS 的设置 (p. 7)中的步骤。启动实例之后，您可以使用实例来运行任务。

启动容器实例

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中，选择要使用的区域。
3. 从控制台控制面板中，选择 Launch Instance。
4. 在选择一个 Amazon 系统映像(AMI) 页面上，完成以下步骤：

   a. 选择社区 AMI。
   b. 为您的容器实例选择 AMI。您可以选择经 Amazon ECS 优化的 AMI 或其他操作系统，如 CoreOS 或 Ubuntu。如果未选择经 Amazon ECS 优化的 AMI，则必须按照安装 Amazon ECS 容器代理 (p. 198)中的过程操作。

Note

有关 Amazon ECS 特定 CoreOS 安装说明的更多信息，请参阅使用 AWS EC2 Container Service 运行 CoreOS 容器 (Linux)。

有关最新的经 Amazon ECS 优化的 AMI，请参阅 如何启动最新的 Amazon ECS-Optimized Amazon Linux AMI (p. 137)。

5. 在 Choose an Instance Type (选择实例类型) 页面上，您可以选择实例的硬件配置。默认情况下会选择 `t2.micro` 实例类型。您选择的实例类型决定了可用于运行您的任务的资源。

   完成后，选择下一步: 配置实例详细信息。

6. 在配置实例详细信息页面上，完成以下步骤：

   a. 根据要添加到集群中的容器实例数量设置 Number of instances 字段。

   b. (可选) 要使用 Spot 实例，对于购买选项，请选中请求 Spot 实例旁边的复选框。您还需要设置与 Spot 实例相关的其他字段。有关更多信息，请参阅 Spot 实例请求。

      Note

      如果使用 Spot 实例时看到 `Not available` 消息，则需要选择其他实例类型。

   c. 对于网络，选择用于启动容器实例的 VPC。

   d. 对于 Subnet，选择要使用的子网，或保留默认选项在任意可用区中选择默认子网。

   e. 根据是否要使实例可从公共 Internet 访问，设置自动分配公有 IP 字段。如果实例应可从 Internet 进行访问，请确保自动分配公有 IP 字段设置为启用。如果不是，请将此字段设置为禁用。

      Note

      容器实例需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的容器实例完成。
      有关接口 VPC 终端节点的更多信息，请参阅Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink) (p. 337)。
      如果您没有配置接口 VPC 终端节点并且您的容器实例没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关和本指南中的 HTTP 代理配置 (p. 245)。有关更多信息，请参阅Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters (p. 555)。

   f. 选择您在`ecsInstanceRole`中为自己的容器实例创建的 IAM roleAmazon ECS 的设置 (p. 7) 值。

      Important

      如果未使用适当的 IAM 权限启动容器实例，则 Amazon ECS 代理无法连接到集群。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

   g. （可选）使用用户数据（如 Amazon ECS 容器代理配置 (p. 217)中的代理环境变量）配置 Amazon ECS 容器实例。Amazon EC2 用户数据脚本仅在实例首次启动时执行一次。以下是用户数据的常用示例：

      - 默认情况下，您的容器实例将启动到您的默认集群中。要在非默认集群中启动，请选择 Advanced Details 列表。然后，将以下脚本粘贴到 User data 字段中，将 *`your_cluster_name`* 替换为您的集群的名称。

        ```
        #!/bin/bash
        echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
        ```

      - 如果 Amazon S3 中有 `ecs.config` 文件并且启用了对容器实例角色的 Amazon S3 只读访问权限，请选择高级详细信息列表。然后，将以下脚本粘贴到用户数据字段中，并将 *`your_bucket_name`* 替换为您的存储桶的名称以安装 AWS CLI 和在启动时写入您的配置文件。

Note

有关此配置的更多信息，请参阅将容器实例配置存储在 Amazon S3 中 (p. 227)。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

- 使用 `ECS_CONTAINER_INSTANCE_TAGS` 配置参数为您的容器实例指定标签。这只创建与 Amazon ECS 关联的标签，无法使用 Amazon EC2 API 列出这些标签。

  Important

  如果您使用 Amazon EC2 Auto Scaling 组启动容器实例，则应使用 ECS_CONTAINER_INSTANCE_TAGS 代理配置参数添加标签。这是将标签添加到使用 Auto Scaling 组所启动的 Amazon EC2 实例的方式决定的。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- 为您的容器实例指定标签，然后使用 `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` 配置参数将其从 Amazon EC2 传播到 Amazon ECS

  下面是一个用户数据脚本示例，该脚本将传播与容器实例关联的标签，以及向名为 `your_cluster_name` 的集群注册容器实例：

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
EOF
```

有关更多信息，请参阅 使用 Amazon EC2 用户数据引导启动容器实例 (p. 168)。

h. 选择 Next: Add Storage。

7. 在 Add Storage (添加存储) 页面上，为您的容器实例配置存储。

   如果您使用的是 Amazon ECS-optimized Amazon Linux 2 AMI，您的实例将配置单个 30GiB 卷，用于在操作系统和 Docker 之间共享。

   如果您使用的是经 Amazon ECS 优化的 AMI，您的实例将配置两个卷。Root (根) 卷适合操作系统使用，第二个 Amazon EBS 卷（已挂载到 `/dev/xvdcz`）适合 Docker 使用。

   您可以选择增大或减小实例的卷大小以满足您的应用程序需求。

   在配置卷后，选择下一步: 添加标签。

8. 在添加标签页面上，通过提供键和值组合为容器实例指定标签。选择添加其他标签以向您的容器实例添加多个标签。有关资源标签的更多信息，请参阅资源和标签 (p. 330)。

   完成时选择 Next: Configure Security Group。

9. 在配置安全组页面上，使用安全组为容器实例定义防火墙规则。这些规则指定哪些传入的网络流量可传输到您的容器实例。所有其他的流量将被忽略。按如下所示选择或创建安全组，然后选择审核和启动。

10. 在 Review Instance Launch 页面上的 Security Groups 下，您将看到向导为您创建并选择了安全组。相反，请选择您在Amazon ECS 的设置 (p. 70)中使用以下步骤创建的安全组：

   a. 选择 Edit security groups。
   b. 在 Configure Security Group 页面上，选择 Select an existing security group 选项。
   c. 从现有安全组列表中选择您为容器实例创建的安全组，然后选择 Review and Launch。
11. 在 Review Instance Launch 页面上，选择 Launch。
12. 在 Select an existing key pair or create a new key pair 对话框中，选择 Choose an existing key pair，然后选择您在开始设置时创建的密钥对。

    准备好后，选中确认字段，然后选择 Launch Instances。
13. 确认页面会让您知道自己的实例已启动。选择 View Instances 以关闭确认页面并返回控制台。
14. 在 Instances (实例) 屏幕上，您可以查看您实例的状态。启动实例只需很短的时间。启动实例时，其初始状态为 `pending`。实例启动后，其状态变为 `running`，并且会收到一个公有 DNS 名称。如果 Public DNS 列已隐藏，请依次选择 Show/Hide 和 Public DNS。

使用 Amazon EC2 用户数据引导启动容器实例

在启动 Amazon ECS 容器实例时，您可以选择将用户数据传递到实例。数据可以用于执行常见的自动配置任务，甚至用于在实例启动时运行脚本。对于 Amazon ECS，最常见的用户数据使用案例是将配置信息传递到 Docker 守护程序和 Amazon ECS 容器实例。

您可以将多类用户数据传递到 Amazon EC2，其中包括云 boothook、Shell 脚本和 `cloud-init` 指令。有关这些和其他格式类型的更多信息，请参阅 Cloud-Init 文档。

您可以将此用户数据传递到启动 Amazon ECS 容器实例 (p. 165)的Step 6.g (p. 166)的 Amazon EC2 启动向导中。

主题

- Amazon ECS 容器代理 (p. 168)
- Docker 守护程序 (p. 169)
- cloud-init-per 实用工具 (p. 169)
- 使用 MIME 分段存档指定多个用户数据块 (p. 170)
- 示例容器实例用户数据配置脚本 (p. 171)

Amazon ECS 容器代理

当容器代理启动时，Amazon ECS-optimized AMI 的 Linux 变体将在 `/etc/ecs/ecs.config` 文件中查找代理配置数据。您可以在启动时使用 Amazon EC2 用户数据指定此配置数据。有关可用 Amazon ECS 容器代理配置变量的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

若要仅设置一个代理配置变量（如集群名称），请使用 echo 将该变量复制到配置文件：

```
#!/bin/bash
echo "ECS_CLUSTER=MyCluster" >> /etc/ecs/ecs.config
```

如果您有多个变量要写入到 `/etc/ecs/ecs.config`，请使用以下 `heredoc` 格式。此格式会将以 cat 和 `EOF` 开头的行之间的所有内容写入到配置文件。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENGINE_AUTH_TYPE=docker
```

```
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
ECS_LOGLEVEL=debug
EOF
```

Docker 守护程序

您可以使用 Amazon EC2 用户数据指定 Docker 守护程序配置信息，但此配置数据必须在 Docker 守护程序启动之前写入。在启动过程中，`cloud-boothook` 用户数据格式早于用户数据 Shell 脚本执行。有关配置选项的更多信息，请参阅 Docker 守护程序文档。

默认情况下，`cloud-boothook` 用户数据在每个实例启动时运行，因此您必须创建一个机制来阻止 boothook 运行多次。提供了 cloud-init-per 实用程序来以此方式控制 boothook 频率。有关更多信息，请参阅cloud-init-per 实用工具 (p. 169)。

在以下示例中，`--foo bar` 选项已附加到 Docker 守护程序配置文件 `/etc/sysconfig/docker` 中的所有现有选项。

```
#cloud-boothook
cloud-init-per once docker_options echo 'OPTIONS="${OPTIONS} --foo bar"' >> /etc/sysconfig/
docker
```

要向一个文件写入多个行，请使用以下 `heredoc` 格式来实现相同的目标：

```
#cloud-boothook
cloud-init-per instance docker_options cat <<'EOF' >> /etc/sysconfig/docker
OPTIONS="${OPTIONS} --foo bar"
HTTP_PROXY=http://proxy.example.com:80/
EOF
```

cloud-init-per 实用工具

cloud-init-per 实用工具由 `cloud-init` 程序包提供，用于帮助您为实例创建以指定频率运行的 boothook 命令。

cloud-init-per 实用程序的语法如下所示：

```
cloud-init-per frequency name cmd [ arg1 [ arg2 [ ... ] ]
```

`frequency`

boothook 应运行的频率。

- 指定 `once` 可不再运行 (即使使用新的实例 ID)。
- 指定 `instance` 可在每个新实例首次启动时运行。例如，如果您在 boothook 运行后从实例创建 AMI，boothook 仍将在从该 AMI 启动的后续实例上再次运行。
- 指定 `always` 可在每次启动时运行。

`name`

要包含到在 boothook 运行时写入的信号灯文件路径中的名称。信号灯文件将写入到 `/var/lib/cloud/instances/`*`instance_id`*`/sem/bootper.`*`name`*`.instance`。

`cmd`

boothook 应执行的命令和参数。

在以下示例中，命令 `echo 'OPTIONS="${OPTIONS} --foo bar"' >> /etc/sysconfig/docker` 只会执行一次。将写入包含自身名称的信号灯文件。

```
#cloud-boothook
cloud-init-per once docker_options echo 'OPTIONS="${OPTIONS} --foo bar"' >> /etc/sysconfig/
docker
```

该信号灯文件将记录命令的退出代码和执行命令时的 UNIX 时间戳。

```
[ec2-user ~]$ cat /var/lib/cloud/instances/i-0c7f87d7611b2165e/sem/
bootper.docker_options.instance
```

输出：

```
0 1488410363
```

使用 MIME 分段存档指定多个用户数据块

您可以将多个用户数据块合并到一个名为 MIME 分段文件的用户数据块中。例如，您可能希望将配置 Docker 守护程序的云 boothook 与为 Amazon ECS 容器代理写入配置信息的用户数据 Shell 脚本合并。

MIME 分段文件包含以下组成部分：

- 内容类型和段边界声明：`Content-Type: multipart/mixed; boundary="==BOUNDARY=="`
- MIME 版本声明：`MIME-Version: 1.0`
- 一个或多个用户数据块，其包含以下组成部分：
  - 开口边界，表示用户数据块的开头：`--==BOUNDARY==`
  - 数据块的内容类型声明：`Content-Type: text/cloud-boothook; charset="us-ascii"`。有关内容类型的更多信息，请参阅 Cloud-Init 文档。
  - 用户数据的内容，例如，Shell 命令或 `cloud-init` 指令的列表
- 封闭边界，表示 MIME 分段文件的结尾：`--==BOUNDARY==--`

Example MIME 分段文件

此示例 MIME 分段文件会将 Docker 基本设备尺寸配置为 20 GiB，并将 Amazon ECS 容器代理配置为将实例注册到名为 `my-ecs-cluster` 的集群。

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/cloud-boothook; charset="us-ascii"

# Set Docker daemon options
cloud-init-per once docker_options echo 'OPTIONS="${OPTIONS} --foo bar"' >> /etc/sysconfig/
docker

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Set any ECS agent configuration options
echo "ECS_CLUSTER=my-ecs-cluster" >> /etc/ecs/ecs.config
```

```
--==BOUNDARY==--
```

示例容器实例用户数据配置脚本

下面的示例用户数据脚本在启动时配置 Amazon ECS 容器实例。

具有 Amazon EFS 文件系统的 Amazon ECS-Optimized Amazon Linux AMI 容器实例

该示例用户数据脚本将从 Amazon ECS-optimized Amazon Linux AMI 启动的实例配置为使用现有的 Amazon EFS 文件系统。有关更多信息，请参阅 Tutorial: Using Amazon EFS File Systems with Amazon ECS (p. 573)。

该脚本执行以下操作：

- 安装 `nfs-utils` 程序包，这会安装 NFS 客户端。
- 在 `/efs` 处为该 NFS 文件系统创建一个挂载目录。
- 在 `/etc/fstab` 文件中为该文件系统创建一个挂载条目，然后挂载该文件系统。
- 将集群名称 *`default`* 写入 Amazon ECS 代理配置文件。

您可以将该脚本用于自己的容器实例，但前提是它们是从 Amazon ECS-optimized Amazon Linux AMI 启动的。如果您使用的不是 `default` 集群，请务必替换配置文件中的 `ECS_CLUSTER=`*`default`* 行，以指定您自己的集群名称。有关启动容器实例的更多信息，请参阅启动 Amazon ECS 容器实例 (p. 165)。

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/cloud-boothook; charset="us-ascii"

# Install nfs-utils
cloud-init-per once yum_update yum update -y
cloud-init-per once install_nfs_utils yum install -y nfs-utils

# Create /efs folder
cloud-init-per once mkdir_efs mkdir /efs

# Mount /efs
cloud-init-per once mount_efs echo -e 'fs-abcd1234.efs.us-east-1.amazonaws.com:/ /efs nfs4
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 0 0' >> /etc/fstab
mount -a

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Set any ECS agent configuration options
echo "ECS_CLUSTER=default" >> /etc/ecs/ecs.config

--==BOUNDARY==--
```

具有 `systemd`的 Ubuntu 容器实例

该示例用户数据脚本将 Ubuntu 16.04 实例配置为：

- 安装 Docker.

- 为任务的 IAM 角色创建所需的 iptables 规则。
- 为 Amazon ECS 容器代理创建所需的目录。
- 写入 Amazon ECS 容器代理配置文件。
- 写入 systemd 单元文件以监视代理。
- 启用并启动 systemd 单元。

您可以将该脚本用于自己的容器实例，但前提是它们是从 Ubuntu 16.04 AMI 启动的。如果您使用的不是 `default` 集群，请务必替换配置文件中的 `ECS_CLUSTER=default` 行，以指定您自己的集群名称。有关启动容器实例的更多信息，请参阅启动 Amazon ECS 容器实例 (p. 165)。

```
#!/bin/bash
# Install Docker
apt-get update -y && apt-get install -y docker.io

# Set iptables rules
echo 'net.ipv4.conf.all.route_localnet = 1' >> /etc/sysctl.conf
sysctl -p /etc/sysctl.conf
iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-destination
 127.0.0.1:51679
iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports
 51679

# Write iptables rules to persist after reboot
iptables-save > /etc/iptables/rules.v4

# Create directories for ECS agent
mkdir -p /var/log/ecs /var/lib/ecs/data /etc/ecs

# Write ECS config file
cat << EOF > /etc/ecs/ecs.config
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
EOF

# Write systemd unit file
cat << EOF > /etc/systemd/system/docker-container@ecs-agent.service
[Unit]
Description=Docker Container %I
Requires=docker.service
After=docker.service

[Service]
Restart=always
ExecStartPre=-/usr/bin/docker rm -f %i
ExecStart=/usr/bin/docker run --name %i \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
ExecStop=/usr/bin/docker stop %i

[Install]
WantedBy=default.target
```

```
EOF

systemctl enable docker-container@ecs-agent.service
systemctl start docker-container@ecs-agent.service
```

具有 systemd 和 SELinux 的 CentOS 容器实例

该示例用户数据脚本将启用了 SELinux 的 CentOS 7 实例配置为：

- 安装 Docker.
- 为任务的 IAM 角色创建所需的 iptables 规则。
- 为 Amazon ECS 容器代理创建所需的目录。
- 写入 Amazon ECS 容器代理配置文件。
- 写入 systemd 单元文件以监视代理。
- 启用并启动 systemd 单元。

> **Note**
>
> 下面的 systemd 单元文件中的 docker run 命令包含 SELinux 所需的修改，其中含有 --privileged 标记和用于卷挂载的 :Z 后缀。

您可以将该脚本用于自己的容器实例（前提是它们是从 CentOS 7 AMI 启动的），但请记得替换配置文件中的 ECS_CLUSTER=default 行来指定自己的集群名称（如果您使用的不是 default 集群）。有关启动容器实例的更多信息，请参阅启动 Amazon ECS 容器实例 (p. 165)。

```
#!/bin/bash
# Install Docker
yum install -y docker

# Set iptables rules
echo 'net.ipv4.conf.all.route_localnet = 1' >> /etc/sysctl.conf
sysctl -p /etc/sysctl.conf
iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-destination
 127.0.0.1:51679
iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports
 51679

# Write iptables rules to persist after reboot
iptables-save > /etc/sysconfig/iptables

# Create directories for ECS agent
mkdir -p /var/log/ecs /var/lib/ecs/data /etc/ecs

# Write ECS config file
cat << EOF > /etc/ecs/ecs.config
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
EOF

# Write systemd unit file
cat << EOF > /etc/systemd/system/docker-container@ecs-agent.service
[Unit]
Description=Docker Container %I
Requires=docker.service
```

```
After=docker.service

[Service]
Restart=always
ExecStartPre=-/usr/bin/docker rm -f %i
ExecStart=/usr/bin/docker run --name %i \
--privileged \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log:Z \
--volume=/var/lib/ecs/data:/data:Z \
--volume=/etc/ecs:/etc/ecs \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
ExecStop=/usr/bin/docker stop %i

[Install]
WantedBy=default.target
EOF

systemctl enable docker-container@ecs-agent.service
systemctl start docker-container@ecs-agent.service
```

默认 Windows 用户数据

该示例用户数据脚本显示在使用集群创建向导 (p. 38) 时 Windows 容器实例接收的默认用户数据。下面的脚本执行以下操作：

- 将集群名称设置为 `windows`。
- 启用任务的 IAM 角色。
- 将 `json-file` 以及 `awslogs` 设置为可用的日志记录驱动程序。

您可以将该脚本用于自己的容器实例（前提是它们是从 Amazon ECS-optimized Windows AMI 启动的），但请记得替换 `-Cluster windows` 行来指定自己的集群名称（如果您使用的不是名为 `windows` 的集群）。

```
<powershell>
Initialize-ECSAgent -Cluster windows -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Windows 代理安装用户数据

该示例用户数据脚本将在使用 Windows_Server-2016-English-Full-Containers AMI 启动的实例上安装 Amazon ECS 容器代理。该脚本已根据 Amazon ECS 容器代理的 GitHub 存储库 README 页面中的代理安装说明进行了调整。

> **Note**
>
> 此脚本作为示例分享。使用 Amazon ECS-optimized Windows AMI 的 Windows 容器更容易上手。有关更多信息，请参阅创建集群 (p. 38)。

您可以将该脚本用于自己的容器实例（前提是它们是使用 Windows_Server-2016-English-Full-Containers AMI 版本启动的）。请记得替换 `windows` 行来指定自己的集群名称（如果您使用的不是名为 `windows` 的集群）。

```
<powershell>
# Set up directories the agent uses
New-Item -Type directory -Path ${env:ProgramFiles}\Amazon\ECS -Force
```

```
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS\data -Force
# Set up configuration
$ecsExeDir = "${env:ProgramFiles}\Amazon\ECS"
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", "windows", "Machine")
[Environment]::SetEnvironmentVariable("ECS_LOGFILE", "${env:ProgramData}\Amazon\ECS\log
\ecs-agent.log", "Machine")
[Environment]::SetEnvironmentVariable("ECS_DATADIR", "${env:ProgramData}\Amazon\ECS\data",
 "Machine")
# Download the agent
$agentVersion = "latest"
$agentZipUri = "https://s3.amazonaws.com/amazon-ecs-agent/ecs-agent-windows-
$agentVersion.zip"
$zipFile = "${env:TEMP}\ecs-agent.zip"
Invoke-RestMethod -OutFile $zipFile -Uri $agentZipUri
# Put the executables in the executable directory.
Expand-Archive -Path $zipFile -DestinationPath $ecsExeDir -Force
Set-Location ${ecsExeDir}
# Set $EnableTaskIAMRoles to $true to enable task IAM roles
# Note that enabling IAM roles will make port 80 unavailable for tasks.
[bool]$EnableTaskIAMRoles = $false
if (${EnableTaskIAMRoles}) {
  $HostSetupScript = Invoke-WebRequest https://raw.githubusercontent.com/aws/amazon-ecs-
agent/master/misc/windows-deploy/hostsetup.ps1
  Invoke-Expression $($HostSetupScript.Content)
}
# Install the agent service
New-Service -Name "AmazonECS" `
        -BinaryPathName "$ecsExeDir\amazon-ecs-agent.exe -windows-service" `
        -DisplayName "Amazon ECS" `
        -Description "Amazon ECS service runs the Amazon ECS agent" `
        -DependsOn Docker `
        -StartupType Manual
sc.exe failure AmazonECS reset=300 actions=restart/5000/restart/30000/restart/60000
sc.exe failureflag AmazonECS 1
Start-Service AmazonECS
</powershell>
```

弹性网络接口中继

每个使用 `awsvpc` 网络模式的 Amazon ECS 任务都会接收到自己的弹性网络接口 (ENI)，该网络接口附加到托管它的容器实例。可以附加到 Amazon EC2 实例的网络接口的数量有默认限制，并且主网络接口计为一个。例如，默认情况下，最多可将三个 ENI 附加到一个 `c5.large` 实例。实例的主网络接口计为一个，因此您可以向该实例再附加 2 个 ENI。由于每个使用 `awsvpc` 网络模式的任务均需一个 ENI，因此，您通常只能在该实例类型上运行 2 个此类任务。

Amazon ECS 支持使用受支持的 Amazon EC2 实例类型启动已增加 ENI 密度的容器实例。当您使用这些实例类型并选择使用 `awsvpcTrunking` 账户设置时，其他 ENI 将在新启动的容器实例上可用。此配置允许您使用 `awsvpc` 网络模式在每个容器实例上放置更多任务。通过使用此功能，启用了 `awsvpcTrunking` 的 `c5.large` 实例的 ENI 限制提高了 12。容器实例将具有主网络接口，而 Amazon ECS 将创建一个“中继”网络接口并将此接口附加到容器实例。因此，此配置允许您在容器实例上启动 10 个任务，而不是当前的两个任务。

中继网络接口完全由 Amazon ECS 管理，并且会在您从集群中终止或取消注册容器实例时被删除。有关更多信息，请参阅`awsvpc` 网络模式的任务联网 (p. 95)。

ENI 中继注意事项

在使用 ENI 中继功能时，需要考虑几个事项。

- 仅经过 Amazon ECS 优化的 AMI 的 Linux 变体或其他 Amazon Linux 变体（具有 `1.28.1` 版本或更高版本的容器代理以及 `1.28.1-2` 版本或更高版本的 ecs-init 程序包）支持增加的 ENI 限制。如果您使用经过 Amazon ECS 优化的 AMI 的最新 Linux 变体，则将满足这些要求。目前不支持 Windows 容器。
- 仅在选择使用 `awsvpcTrunking` 后启动的新 Amazon EC2 实例将收到增加的 ENI 限制和中继网络接口。无论采取何种措施，以前启动的实例都不会收到这些功能。
- 您的 Amazon ECS 任务必须使用 `awsvpc` 网络模式和 EC2 启动类型。不管启动的实例数量如何，使用 Fargate 启动类型的任务都始终会收到专用的 ENI，因此不需要此功能。
- 在启动新的容器实例时，实例将转换为 `REGISTERING` 状态，同时为实例预配置中继弹性网络接口。如果注册失败，则实例将转换为 `REGISTRATION_FAILED` 状态。您可以描述容器实例并查看 `statusReason` 参数以了解失败原因。
- 一旦终止容器实例，该实例就会转换为 `DEREGISTERING` 状态，同时取消预配置中继弹性网络接口。随后，实例将转换为 `INACTIVE` 状态。
- 如果停止并重新启动具有增加的 ENI 限制的公有子网中的容器实例，则实例将丢失其公有 IP 地址，并且容器代理将丢失其连接。

使用具有增加的 ENI 限制的容器实例

在启动具有增加的 ENI 限制的容器实例之前，必须完成以下先决条件。

- 必须创建 Amazon ECS 的服务相关角色。Amazon ECS 服务相关角色向 Amazon ECS 提供代表您调用其他 AWS 服务的权限。此角色是在创建集群时 (或者在 AWS 管理控制台 中创建或更新服务时) 自动为您创建的。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色 (p. 392)。您也可以使用以下 AWS CLI 命令创建服务相关角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的账户或容器实例 IAM 角色必须选择使用 `awsvpcTrunking` 账户设置。可通过以下方式执行此任务：
  - 任何用户都可以使用 PutAccountSettingDefault API 来选择账户上的所有 IAM 用户和角色
  - 根用户可以使用 PutAccountSetting API 来选择将在集群上注册实例的 IAM 用户或容器实例角色
  - 在某个实例注册到集群之前，当 PutAccountSetting API 运行在该实例上时，容器实例角色可以选择使用自身

  有关更多信息，请参阅 账户设置 (p. 120)。

在满足先决条件后，您可以使用受支持的 Amazon EC2 实例类型之一来启动新容器实例，并且实例将具有增加的 ENI 限制。有关受支持实例类型的列表，请参阅支持的 Amazon EC2 实例类型 (p. 178)。容器实例必须具有 `1.28.1` 版本或更高版本的容器代理以及 `1.28.1-2` 版本或更高版本的 ecs-init 程序包。如果您使用经过 Amazon ECS 优化的 AMI 的最新 Linux 变体，则将满足这些要求。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。

使用控制台为您账户上的所有 IAM 用户和角色选择增加的 ENI 限制。

1. 以账户的根用户身份打开位于 https://console.aws.amazon.com/ecs/ 的 Amazon ECS 控制台。
2. 在屏幕顶部的导航栏中，选择要增加 ENI 限制的区域。
3. 从控制面板中选择账户设置。
4. 对于 IAM 用户或角色，确保选择了您的根用户或容器实例 IAM 角色。
5. 对于 AWSVPC 中继，请选中复选框。完成后，选择保存。

   Important

   IAM 用户和 IAM 角色需要 `ecs:PutAccountSetting` 权限来执行此操作。
6. 在确认屏幕上，选择 Confirm (确认) 以保存选择。

使用命令为您账户上的所有 IAM 用户和角色选择增加的 ENI 限制。

账户上的任意用户可以使用下列命令之一修改账户中所有 IAM 用户或角色的默认账户设置。这些更改将应用于整个 AWS 账户，除非一个 IAM 用户或角色显式覆盖自己的这些设置。

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name awsvpcTrunking --value enabled --region us-
east-1
```

- Write-ECSAccountSettingDefault (适用于 Windows PowerShell 的 AWS 工具)

```
Write-ECSAccountSettingDefault -Name awsvpcTrunking -Value enabled -Region us-east-1 -
Force
```

使用命令行，以根用户身份为 IAM 用户或容器实例 IAM 角色选择增加的 ENI 限制

账户的根用户可以使用以下命令之一，并在请求中指定委托人 IAM 用户或容器实例 IAM 角色的 ARN 来修改账户设置。

- put-account-setting (AWS CLI)

  以下示例用于修改特定 IAM 用户的账户设置：

```
aws ecs put-account-setting --name awsvpcTrunking --value enabled --principal-arn
 arn:aws:iam::aws_account_id:user/userName --region us-east-1
```

  以下示例用于修改特定容器实例 IAM 角色的账户设置：

```
aws ecs put-account-setting --name awsvpcTrunking --value enabled --principal-arn
 arn:aws:iam::aws_account_id:role/ecsInstanceRole --region us-east-1
```

- Write-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

  以下示例用于修改特定 IAM 用户的账户设置：

```
Write-ECSAccountSetting -Name awsvpcTrunking -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/userName -Region us-east-1 -Force
```

  以下示例用于修改特定容器实例 IAM 角色的账户设置：

```
Write-ECSAccountSetting -Name awsvpcTrunking -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:role/ecsInstanceRole -Region us-east-1 -Force
```

使用 AWS CLI 查看具有增加的 ENI 限制的容器实例

每个容器实例都有一个默认网络接口，该接口称为中继网络接口。使用以下命令通过查询 `ecs.awsvpc-trunk-id` 属性来列出具有增加的 ENI 限制的容器实例，这表明它具有中继网络接口。

- list-attributes (AWS CLI)

```
aws ecs list-attributes --target-type container-instance --attribute-name ecs.awsvpc-
trunk-id --region us-east-1
```

- Get-ECSAttributeList (适用于 Windows PowerShell 的 AWS 工具)

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-id -
Region us-east-1
```

支持的 Amazon EC2 实例类型

下面显示了受支持的 Amazon EC2 实例类型，以及在选择使用 `awsvpcTrunking` 账户设置之前和之后，可以在每个实例类型上启动使用 `awsvpc` 网络模式的任务的数目。对于每个实例类型上的弹性网络接口 (ENI) 限制，将一个添加到当前任务限制中，因为主网络接口将计入限制，并将两个添加到新任务限制中，因为主网络接口和中继网络实例将再次计入限制。

实例类型	每个实例的当前任务限制	每个实例的新任务限制
a1 instance family		
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20
a1.2xlarge	3	40
a1.4xlarge	7	60
c5 instance family		
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60
c5.18xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.18xlarge	14	120
c5n.large	2	10
c5n.xlarge	3	20
c5n.2xlarge	3	40
c5n.4xlarge	7	60
c5n.9xlarge	7	60

实例类型	每个实例的当前任务限制	每个实例的新任务限制
c5n.18xlarge	14	120
m5 instance family		
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.12xlarge	7	60
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40
m5a.4xlarge	7	60
m5a.12xlarge	7	60
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.12xlarge	7	60
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.12xlarge	7	60
m5d.24xlarge	14	120
p3 instance family		
p3.2xlarge	3	40
p3.8xlarge	7	60
p3.16xlarge	7	120
p3dn.24xlarge	14	120

实例类型	每个实例的当前任务限制	每个实例的新任务限制
r5 instance family		
r5.large	2	10
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40
r5a.4xlarge	7	60
r5a.12xlarge	7	60
r5a.24xlarge	14	120
r5ad.large	2	10
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60
r5ad.12xlarge	7	60
r5ad.24xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.12xlarge	7	60
r5d.24xlarge	14	120

连接到您的容器实例

要在您的实例上执行基本管理任务 (例如，更新或安装软件或访问诊断日志)，请使用 SSH 连接到实例。要使用 SSH 连接到您的实例，您的容器实例必须满足以下先决条件：

- 容器实例需要外部网络访问权限才能通过 SSH 进行连接。如果容器实例正在私有 VPC 中运行，则它们需要 SSH 堡垒实例来提供此访问权限。有关更多信息，请参阅安全地连接到在私有 Amazon VPC 中运行的 Linux 实例博文。

- 容器实例必须已使用有效的 Amazon EC2 密钥对启动。Amazon ECS 容器实例没有密码，您可以通过 SSH 使用密钥对进行登录。如果您在启动实例时未指定密钥对，则无法连接到实例。
- SSH 使用端口 22 进行通信。端口 22 必须在您的容器实例安全组中开放，以便您使用 SSH 连接到您的实例。

  Note

  Amazon ECS 控制台首次运行体验将为您的容器实例创建安全组，而无需端口 22 上的入站访问。如果您的容器实例是通过控制台首次运行体验启动的，请在用于这些实例的安全组上添加对端口 22 的入站访问权限。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的授权网络访问您的实例。

连接到您的容器实例

1. 查找您的容器实例的公共 IP 或 DNS 地址。
   a. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
   b. 选择托管您的容器实例的集群。
   c. 在 Cluster 页面上，选择 ECS Instances。
   d. 在 Container Instance 列上，选择要连接到的容器实例。
   e. 在 Container Instance 页面上，记录您的实例的 Public IP 或 Public DNS。
2. 查找您的容器实例 AMI 的默认用户名。使用Amazon ECS-optimized AMI 启动的实例的用户名为 `ec2-user`。对于 Ubuntu AMI，默认用户名为 `ubuntu`。对于 CoreOS，默认用户名为 `core`。
3. 如果您使用的是 macOS 或 Linux 计算机，请使用以下命令连接到您的实例，并替换您的私有密钥的路径和实例的公共地址：

```
$ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
```

   有关使用 Windows 计算机的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的使用 PuTTY 从 Windows 连接到您的 Linux 实例。

   Important

   有关在连接到实例时遇到的任何问题的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排查实例的连接问题。

将 CloudWatch Logs 用于容器实例

您可以将容器实例配置为向 CloudWatch Logs 发送日志信息。这使您能够在一个方便位置查看容器实例中的不同日志。本主题可帮助您在通过 Amazon ECS-optimized Amazon Linux AMI 启动的容器实例上开始使用 CloudWatch Logs。

有关将任务的容器日志发送到 CloudWatch Logs 的信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。有关 CloudWatch Logs 的更多信息，请参阅 Amazon CloudWatch 用户指南 中的监控日志文件。

主题

- CloudWatch Logs IAM 策略 (p. 182)
- 安装 CloudWatch Logs 代理 (p. 182)
- 配置并启动 CloudWatch Logs 代理 (p. 183)
- 查看 CloudWatch Logs (p. 185)
- 在启动时使用用户数据配置 CloudWatch Logs (p. 186)

CloudWatch Logs IAM 策略

您必须先创建 IAM 策略以允许容器实例使用 CloudWatch Logs API 并将该策略附加到 `ecsInstanceRole`，然后您的容器实例才能将日志数据发送到 CloudWatch Logs。

创建 **ECS-CloudWatchLogs** IAM 策略

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Policies。
3. 依次选择创建策略和 JSON。
4. 输入以下策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

5. 选择查看策略。
6. 在查看策略页面上，为名称输入 `ECS-CloudWatchLogs`，然后选择创建策略。

将 **ECS-CloudWatchLogs** 策略附加到 **ecsInstanceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 选择 `ecsInstanceRole`。如果角色不存在，请按照 Amazon ECS 容器实例 IAM 角色 (p. 404)中的过程操作来创建角色。
4. 依次选择权限和附加策略。
5. 要缩小要附加的可用策略的范围，请为筛选条件键入 ECS-CloudWatchLogs。
6. 选中 ECS-CloudWatchLogs 策略左侧的框并选择附加策略。

安装 CloudWatch Logs 代理

在将 `ECS-CloudWatchLogs` 策略添加到您的 `ecsInstanceRole` 后，可以在容器实例上安装 CloudWatch Logs 代理。

> **Note**
>
> 此过程是针对 Amazon ECS-optimized Amazon Linux AMI 编写的，可能不适用于其他操作系统。有关在其他操作系统上安装此代理的信息，请参阅 Amazon CloudWatch 用户指南 中的 CloudWatch Logs 入门。

安装 CloudWatch Logs 代理

- 运行以下命令安装 CloudWatch Logs 代理。

```
[ec2-user ~]$ sudo yum install -y awslogs
```

安装此代理后，继续下一部分以配置此代理。

配置并启动 CloudWatch Logs 代理

CloudWatch Logs 代理配置文件 (`/etc/awslogs/awslogs.conf`) 描述要发送到 CloudWatch Logs 的日志文件。代理配置文件的 `[general]` 部分定义适用于所有日志流的通用配置，并且您可以为要监控的容器实例上的每个文件添加单独的日志流部分。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的 CloudWatch Logs 代理参考。

以下示例配置文件已针对 Amazon ECS-optimized Amazon Linux AMI 进行了配置，并且它为几个常见日志文件提供了日志流：

`/var/log/dmesg`

Linux 内核的消息缓冲区。

`/var/log/messages`

全局系统消息。

`/var/log/docker`

Docker 守护程序日志消息。

> **Note**
>
> 此日志文件路径仅与基于 Amazon Linux AMI 的实例相关。对于 Amazon Linux 2，Docker 日志存储在 `journald` 中，它当前不受 CloudWatch Logs 的支持。

`/var/log/ecs/ecs-init.log`

来自 `ecs-init` 服务的日志消息。

`/var/log/ecs/ecs-agent.log`

来自 Amazon ECS 容器代理的日志消息。

`/var/log/ecs/audit.log`

来自任务凭证提供程序的 IAM 角色的日志消息。

您可以对 Amazon ECS 容器实例使用以下示例文件，但必须将 `{cluster}` 和 `{container_instance_id}` 条目替换为集群名称和每个容器实例的容器实例 ID，以便按集群名称对日志流进行分组，并为每个单独的容器实例分离这些日志流。遵循示例配置文件的过程介绍了替换集群名称和容器实例 ID 占位符的步骤。

```
[general]
state_file = /var/lib/awslogs/agent-state

[/var/log/dmesg]
file = /var/log/dmesg
log_group_name = /var/log/dmesg
log_stream_name = {cluster}/{container_instance_id}
```

```
[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %b %d %H:%M:%S

[/var/log/docker]
file = /var/log/docker
log_group_name = /var/log/docker
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%S.%f

[/var/log/ecs/ecs-init.log]
file = /var/log/ecs/ecs-init.log
log_group_name = /var/log/ecs/ecs-init.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/ecs-agent.log]
file = /var/log/ecs/ecs-agent.log.*
log_group_name = /var/log/ecs/ecs-agent.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/audit.log]
file = /var/log/ecs/audit.log.*
log_group_name = /var/log/ecs/audit.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ
```

配置 CloudWatch Logs 代理

1. 备份现有 CloudWatch Logs 代理配置文件。

```
[ec2-user ~]$ sudo mv /etc/awslogs/awslogs.conf /etc/awslogs/awslogs.conf.bak
```

2. 创建空白配置文件。

```
[ec2-user ~]$ sudo touch /etc/awslogs/awslogs.conf
```

3. 使用文本编辑器打开 `/etc/awslogs/awslogs.conf` 文件，然后将上面的示例文件复制到该文件中。
4. 安装 jq JSON 查询实用程序。

```
[ec2-user ~]$ sudo yum install -y jq
```

5. 查询 Amazon ECS 自检 API 以查找集群名称并其设置为环境变量。

```
[ec2-user ~]$ cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .Cluster')
```

6. 将文件中的 *{cluster}* 占位符替换为您在上一步中设置的环境变量的值。

```
[ec2-user ~]$ sudo sed -i -e "s/{cluster}/$cluster/g" /etc/awslogs/awslogs.conf
```

7. 查询 Amazon ECS 自检 API 操作以查找容器实例 ID 并将其设置为环境变量。

```
[ec2-user ~]$ container_instance_id=$(curl -s http://localhost:51678/v1/metadata | jq -
r '. | .ContainerInstanceArn' | awk -F/ '{print $2}' )
```

8. 将文件中的 *{container_instance_id}* 占位符替换为您在上一步中设置的环境变量的值。

```
[ec2-user ~]$ sudo sed -i -e "s/{container_instance_id}/$container_instance_id/g" /etc/
awslogs/awslogs.conf
```

配置 CloudWatch Logs 代理区域

默认情况下，CloudWatch Logs 代理将数据发送到 `us-east-1` 区域。要将数据发送到其他区域（例如，您的集群所在的区域），可以在 `/etc/awslogs/awscli.conf` 文件中设置该区域。

1. 使用文本编辑器打开 `/etc/awslogs/awscli.conf` 文件。
2. 在 `[default]` 部分中，将 `us-east-1` 替换为要从中查看日志数据的区域。
3. 保存文件并退出您的文本编辑器。

启动 CloudWatch Logs 代理

1. 使用以下命令启动 CloudWatch Logs 代理。
   - Amazon Linux

     ```
     [ec2-user ~]$ sudo service awslogs start
     ```

   - Amazon Linux 2

     ```
     [ec2-user ~]$ sudo systemctl start awslogsd
     ```

2. 使用以下命令确保 CloudWatch Logs 代理在每次系统启动时启动。
   - Amazon Linux

     ```
     [ec2-user ~]$ sudo chkconfig awslogs on
     ```

   - Amazon Linux 2

     ```
     [ec2-user ~]$ sudo systemctl enable awslogsd.service
     ```

查看 CloudWatch Logs

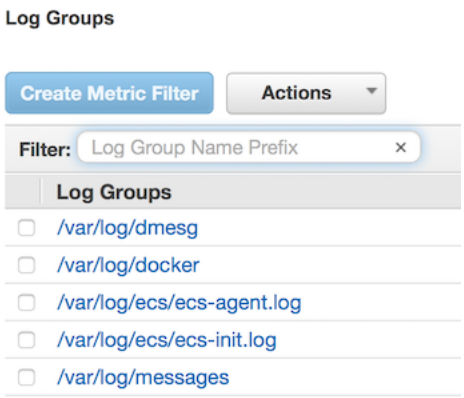
在向容器实例授予将日志发送到 CloudWatch Logs 的适当权限并配置和启动代理后，容器实例应将其日志数据发送到 CloudWatch Logs。您可以在 AWS 管理控制台中查看和搜索这些日志。

> **Note**
>
> 新实例启动可能需要几分钟才能将数据发送到 CloudWatch Logs。

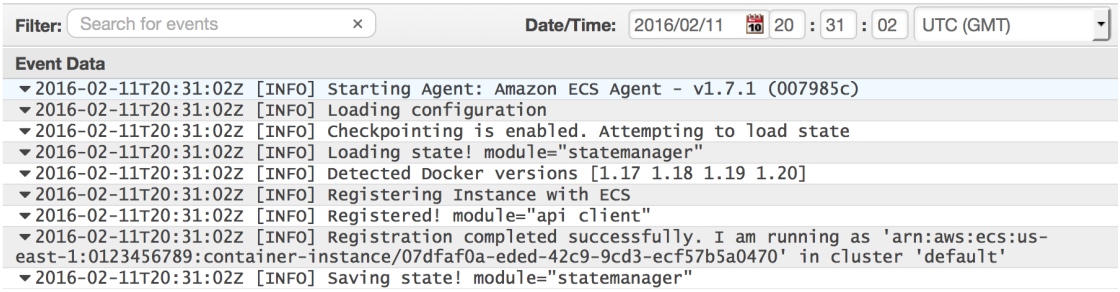
查看您的 CloudWatch Logs 数据

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，选择 Logs。

   您应看到在配置并启动 CloudWatch Logs 代理 (p. 183)中配置的日志组。



3. 选择要查看的日志组。
4. 选择要查看的日志流。这些流由已发送到日志的集群名称和容器实例 ID 标识。



在启动时使用用户数据配置 CloudWatch Logs

当您在 Amazon EC2 中启动 Amazon ECS 容器实例时，您可以选择将用户数据传递到可用于执行常见自动配置任务甚至在实例启动后运行脚本的实例。您可以将多种类型的用户数据传递到实例，包括 Shell 脚本、`cloud-init` 指令和系统服务。您还可以将这些数据以纯文本、文件 (这非常适合通过命令行工具启动实例) 或者 base64 编码文本 (用于 API 调用) 的形式传递到启动向导中。

下面显示的示例用户数据块（请注意，Amazon Linux 2 和 Amazon Linux AMI 变体具有单独的用户数据块）执行以下任务：

- 安装 `awslogs` 程序包，其中包含 CloudWatch Logs 代理
- 安装 jq JSON 查询实用程序
- 编写 CloudWatch Logs 代理的配置文件并配置数据将发送到的区域（容器实例所在的区域）
- 在 Amazon ECS 容器代理启动并将这些值写入 CloudWatch Logs 代理配置文件日志流后，获取集群名称和容器实例 ID
- 启动 CloudWatch Logs 代理
- 将 CloudWatch Logs 代理配置为在每次系统启动时启动

Example Amazon ECS 优化的 Amazon Linux 2 AMI 用户数据

```
Content-Type: multipart/mixed; boundary="==BOUNDARY==="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/usr/bin/env bash
# Install awslogs and the jq JSON parser
```

```
yum install -y awslogs jq

# Inject the CloudWatch Logs configuration file contents
cat > /etc/awslogs/awslogs.conf <<- EOF
[general]
state_file = /var/lib/awslogs/agent-state

[/var/log/dmesg]
file = /var/log/dmesg
log_group_name = /var/log/dmesg
log_stream_name = {cluster}/{container_instance_id}

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %b %d %H:%M:%S

[/var/log/ecs/ecs-init.log]
file = /var/log/ecs/ecs-init.log
log_group_name = /var/log/ecs/ecs-init.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/ecs-agent.log]
file = /var/log/ecs/ecs-agent.log.*
log_group_name = /var/log/ecs/ecs-agent.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/audit.log]
file = /var/log/ecs/audit.log.*
log_group_name = /var/log/ecs/audit.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

EOF

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/usr/bin/env bash
# Write the awslogs bootstrap script to /usr/local/bin/bootstrap-awslogs.sh
cat > /usr/local/bin/bootstrap-awslogs.sh <<- 'EOF'
#!/usr/bin/env bash
exec 2>>/var/log/ecs/cloudwatch-logs-start.log
set -x

until curl -s http://localhost:51678/v1/metadata
do
 sleep 1
done

# Set the region to send CloudWatch Logs data to (the region where the container instance
 is located)
cp /etc/awslogs/awscli.conf /etc/awslogs/awscli.conf.bak
region=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | jq -r .region)
sed -i -e "s/region = .*/region = $region/g" /etc/awslogs/awscli.conf

# Grab the cluster and container instance ARN from instance metadata
cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster')
container_instance_id=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $2}' )

# Replace the cluster name and container instance ID placeholders with the actual values
cp /etc/awslogs/awslogs.conf /etc/awslogs/awslogs.conf.bak
sed -i -e "s/{cluster}/$cluster/g" /etc/awslogs/awslogs.conf
```

```
sed -i -e "s/{container_instance_id}/$container_instance_id/g" /etc/awslogs/awslogs.conf
EOF

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/usr/bin/env bash
# Write the bootstrap-awslogs systemd unit file to /etc/systemd/system/bootstrap-
awslogs.service
cat > /etc/systemd/system/bootstrap-awslogs.service <<- EOF
[Unit]
Description=Bootstrap awslogs agent
Requires=ecs.service
After=ecs.service
Before=awslogsd.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/bootstrap-awslogs.sh

[Install]
WantedBy=awslogsd.service
EOF

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/sh
chmod +x /usr/local/bin/bootstrap-awslogs.sh
systemctl daemon-reload
systemctl enable bootstrap-awslogs.service
systemctl enable awslogsd.service
systemctl start awslogsd.service --no-block

--==BOUNDARY==--
```

Example Amazon ECS 优化的 Amazon Linux AMI 用户数据

```
Content-Type: multipart/mixed; boundary="==BOUNDARY==="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/usr/bin/env bash
# Install awslogs and the jq JSON parser
yum install -y awslogs jq

# Inject the CloudWatch Logs configuration file contents
cat > /etc/awslogs/awslogs.conf <<- EOF
[general]
state_file = /var/lib/awslogs/agent-state

[/var/log/dmesg]
file = /var/log/dmesg
log_group_name = /var/log/dmesg
log_stream_name = {cluster}/{container_instance_id}

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %b %d %H:%M:%S

[/var/log/docker]
file = /var/log/docker
```

```
log_group_name = /var/log/docker
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%S.%f

[/var/log/ecs/ecs-init.log]
file = /var/log/ecs/ecs-init.log
log_group_name = /var/log/ecs/ecs-init.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/ecs-agent.log]
file = /var/log/ecs/ecs-agent.log.*
log_group_name = /var/log/ecs/ecs-agent.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

[/var/log/ecs/audit.log]
file = /var/log/ecs/audit.log.*
log_group_name = /var/log/ecs/audit.log
log_stream_name = {cluster}/{container_instance_id}
datetime_format = %Y-%m-%dT%H:%M:%SZ

EOF

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/usr/bin/env bash
# Set the region to send CloudWatch Logs data to (the region where the container instance
 is located)
region=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | jq -r .region)
sed -i -e "s/region = us-east-1/region = $region/g" /etc/awslogs/awscli.conf

--==BOUNDARY==
Content-Type: text/upstart-job; charset="us-ascii"

#upstart-job
description "Configure and start CloudWatch Logs agent on Amazon ECS container instance"
author "Amazon Web Services"
start on started ecs

script
 exec 2>>/var/log/ecs/cloudwatch-logs-start.log
 set -x

 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the cluster and container instance ARN from instance metadata
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster')
 container_instance_id=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $2}' )

 # Replace the cluster name and container instance ID placeholders with the actual values
 sed -i -e "s/{cluster}/$cluster/g" /etc/awslogs/awslogs.conf
 sed -i -e "s/{container_instance_id}/$container_instance_id/g" /etc/awslogs/awslogs.conf

 service awslogs start
 chkconfig awslogs on
end script
--==BOUNDARY==--
```

如果您已创建 `ECS-CloudWatchLogs` 策略并将其附加到 `ecsInstanceRole` (如 CloudWatch Logs IAM 策略 (p. 182)中所述)，则可以将以上用户数据块添加到您手动启动的任意容器实例。您还可以将其添加到

Auto Scaling 启动配置。使用此用户数据启动的容器实例会在启动之后立即将其日志数据发送到 CloudWatch Logs。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。

容器实例耗尽

有时可能需要删除集群中的容器实例；例如，为了执行系统更新、更新 Docker 守护程序或缩减集群规模。容器实例耗尽允许您从集群中删除容器实例而不影响集群中的任务。

将某个容器实例设置为 `DRAINING` 时，Amazon ECS 将阻止安排放置在该容器实例上的新任务。连接即将耗尽的容器实例上处于 `PENDING` 状态的服务任务将立即停止。如果集群中有可用的容器实例，则在这些容器实例上启动替换服务任务。

容器实例上处于 `RUNNING` 状态的服务任务将停止，并根据服务的部署配置参数 `minimumHealthyPercent` 和 `maximumPercent` 进行替换。

- 如果 `minimumHealthyPercent` 低于 100%，计划程序在替换任务时可以临时忽略 `desiredCount`。例如，`desiredCount` 为四项任务，如果最小值为 50%，则允许计划程序在开始两项新任务之前终止两项现有任务。如果最小值为 100%，那么在替换任务正常运行之前，服务计划程序无法删除现有任务。如果针对未使用负载均衡器的服务的任务处于 `RUNNING` 状态，则认为这些任务正常运行。服务任务的状态如果为 `RUNNING` 且使用负载均衡器，同时该负载均衡器报告托管该服务任务的容器实例运行正常，则该服务任务运行正常。
- `maximumPercent` 参数表示进行任务替换时正在运行的任务数量上限，允许您定义替换批处理大小。例如，如果 `desiredCount` 为四项任务，那么上限 200% 会在停止将要耗尽的四项任务前启动四项新任务 (假设需要进行此操作的集群中有可用资源)。如果上限为 100%，只有在耗尽任务停止后才能启动替换任务。

有关更多信息，请参阅 服务定义参数 (p. 272)。

不属于服务的 `PENDING` 或 `RUNNING` 任务不会受到影响；您必须等待它们完成，或手动停止它们。

如果容器实例中已没有 `RUNNING` 任务，即完成了耗尽 (虽然状态仍为 `DRAINING`)。您可以使用具有 `containerInstance` 参数的 ListTasks 操作进行验证。

如果将容器实例的状态从 `DRAINING` 更改为 `ACTIVE`，Amazon ECS 计划程序可以再次计划该实例的任务。

耗尽实例

您可以使用 UpdateContainerInstancesState API 操作或 update-container-instances-state 命令将容器实例的状态更改为 `DRAINING`。

以下过程说明如何使用 AWS 管理控制台将实例设置为 `DRAINING`。

使用控制台将实例设置为 **`DRAINING`**

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Clusters 并选择集群。
3. 选择 ECS Instances，然后选中容器实例的复选框。
4. 选择 Actions，Drain instances。
5. 实例操作完成后，选择 Done。

容器实例内存管理

当 Amazon ECS 容器代理将容器实例注册到集群时，该代理必须确定容器实例可为任务保留多少内存。由于平台内存开销和系统内核占用的内存，此数量不同于 Amazon EC2 实例所标示的已安装内存量。例

如，`m4.large` 实例具有 8GiB 的已安装内存。但是，当容器实例注册时，这不总是表示确实有 8192 MiB 内存对任务可用。

如果为任务指定 8192 MiB;，而您的任何一个容器实例都没有 8192 MiB 或更高内存可用于满足此需求，则任务无法放置在您的集群中。

您还应该为 Amazon ECS 容器代理以及容器实例上的其他关键系统进程预留一些内存，这样您的任务容器不会争用相同的内存从而可能触发系统故障。 有关更多信息，请参阅预留系统内存 (p. 191)。

Amazon ECS 容器代理使用 Docker `ReadMemInfo()` 函数来查询可用于操作系统的总内存。Linux 和 Windows 都提供了用来确定总内存的命令行实用程序。

Example - 确定 Linux 总内存

free 命令可返回操作系统识别的总内存。

```
$ free -b
```

运行 Amazon ECS-optimized Amazon Linux AMI 的 `m4.large` 实例的示例输出。

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

此实例具有 8373026816 字节的总内存，这表示有 7985MiB 内存可用于任务。

Example - 确定 Windows 总内存

wmic 命令可返回操作系统识别的总内存。

```
C:\> wmic ComputerSystem get TotalPhysicalMemory
```

运行 Amazon ECS-optimized Windows AMI 的 `m4.large` 实例的示例输出。

```
TotalPhysicalMemory
8589524992
```

此实例具有 8589524992 字节的总内存，这表示有 8191MiB 内存可用于任务。

预留系统内存

如果任务占用了容器实例上的全部内存，则任务可能与关键系统进程争用内存，并且可能会触发系统故障。Amazon ECS 容器代理提供名为 `ECS_RESERVED_MEMORY` 的配置变量，您可以使用该变量从分配给您的任务的池中移除指定 MiB 数的内存。这可以有效地为关键系统进程预留该内存。

例如，如果您在容器代理配置文件中指定 `ECS_RESERVED_MEMORY=256`，则代理会为该实例注册总内存减去 256 MiB 后得到的内存量，这 256 MiB 内存无法由 ECS 任务分配。有关代理配置变量以及如何设置这些变量的更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)和使用 Amazon EC2 用户数据引导启动容器实例 (p. 168)。

查看容器实例内存

您可以在 Amazon ECS 控制台中（或者使用 DescribeContainerInstances API 操作）查看容器实例注册的内存量。如果要尝试为任务提供尽可能多的某个具体实例类型的内存，以最大程度地提高资源使用率，则可以观察可用于该容器实例的内存，然后为任务分配该内存量。

查看容器实例内存

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择托管您的容器实例的集群来查看。
3. 选择 ECS Instances (ECS 实例)，然后从 Container Instance (容器实例) 列中选择要查看的容器实例。
4. Resources (资源) 部分显示容器实例的已注册内存和可用内存。

Resources

Resources	Registered	Available
CPU	2048	2048
Memory	7953	7953
Ports	*5 ports*	

Registered (已注册) 内存值是容器实例在首次启动时注册到 Amazon ECS 的内存量；而 Available (可用) 内存值是尚未分配给任务的内存量。

使用 AWS Systems Manager 远程管理容器实例

您可以使用 AWS Systems Manager 中的 Run Command 功能安全地远程管理 Amazon ECS 容器实例的配置。Run Command 提供了一种无需在本地登录实例即可执行常见管理任务的简单方法。您可以通过同时对多个容器实例执行命令来管理集群中的配置更改。Run Command 将报告每个命令的状态和结果。

下面是您可使用 Run Command 执行的任务类型的一些示例：

- 安装或卸载程序包。
- 执行安全更新。
- 清除 Docker 映像。
- 停止或启动服务。
- 查看系统资源。
- 查看日志文件。
- 执行文件操作。

本主题介绍了Amazon ECS-optimized AMI 的 Linux 变体上 Run Command 的基本安装和几个简单的使用案例，但这些内容并不详尽。有关 Run Command 的更多信息，请参阅 AWS Systems Manager 用户指南 中的 AWS Systems Manager Run Command。

主题

- Run Command IAM 策略 (p. 193)
- 在Amazon ECS-Optimized AMI 上安装 SSM 代理 (p. 193)
- 使用 Run Command (p. 194)

Run Command IAM 策略

您必须先附加 `ecsInstanceRole` 访问 Systems Manager API 的 IAM 策略，然后才能利用 Run Command 将命令发送到您的容器实例。以下过程介绍如何将 Systems Manager 托管的策略附加到您的容器实例角色，以便通过此角色启动的实例可使用 Run Command。

将 Systems Manager 策略附加到您的 **`ecsInstanceRole`**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 选择 `ecsInstanceRole`。如果角色不存在，请按照 Amazon ECS 容器实例 IAM 角色 (p. 404)中的过程操作来创建角色。
4. 选择 Permissions 选项卡。
5. 选择附加策略。
6. 要缩小要附加的可用策略的范围，请为筛选条件键入 `SSM`。
7. 在策略列表中，选中 AmazonSSMManagedInstanceCore 旁边的复选框。使用此策略可提供使用 Systems Manager 所需的最低权限。

   有关您可以为 Systems Manager 操作提供的其他策略的信息，请参阅 AWS Systems Manager 用户指南 中的 为 Systems Manager 创建 IAM 实例配置文件
8. 选择 Attach Policy。

在Amazon ECS-Optimized AMI 上安装 SSM 代理

在您将 Systems Manager 策略附加到 `ecsInstanceRole` 之后，可以在容器实例上安装 AWS Systems Manager 代理（SSM 代理）。SSM 代理是一个 Amazon 软件，可以在 Amazon EC2 实例、本地服务器或虚拟机 (VM) 上安装和配置。SSM 代理让 Systems Manager 可以更新、管理和配置这些资源。SSM 代理处理 Run Command 请求，并配置在请求中指定的实例。使用以下过程在Amazon ECS-optimized AMI 容器实例上安装 SSM 代理。

> **Note**
>
> SSM 代理适用于提供了 Systems Manager 的所有区域。（有关受支持区域的列表，请参阅 AWS General Reference 中 AWS Systems Manager 区域和终端节点表主题的区域栏。）每个区域具有其区域特定的下载 URL，但以下命令使用您当前的指定 AWS 区域。有关 SSM 代理的更多信息，请参阅 AWS Systems Manager 用户指南 中的使用 SSM 代理。

在现有的Amazon ECS-optimized AMI 容器实例上手动安装 SSM 代理

1. 连接到您的容器实例。 (p. 180)
2. 运行以下命令安装 SSM 代理 RPM。

```
[ec2-user ~]$ sudo yum install -y amazon-ssm-agent
```

在启动新实例时使用 Amazon EC2 用户数据安装 SSM 代理

- 通过执行启动 Amazon ECS 容器实例 (p. 165)中的过程（但在Step 6.g (p. 166) 中，将以下用户数据脚本复制并粘贴到 User data 字段中）启动一个或多个容器实例。您也可以将此用户数据脚本中的命令添加到执行其他任务（如设置实例要注册到的集群的名称）所需的其他现有脚本。

```
#!/bin/bash
# Install JQ JSON parser
```

```
yum install -y jq

# Get the current region from the instance metadata
region=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document | jq
 -r .region)

# Install the SSM agent RPM
yum install -y https://amazon-ssm-$region.s3.amazonaws.com/latest/linux_amd64/amazon-
ssm-agent.rpm
```

使用 Run Command

在将 Systems Manager 托管策略附加到 `ecsInstanceRole` 并在容器实例上安装 SSM 代理后，您可以开始使用 Run Command 将命令发送到容器实例。有关在您的实例上运行命令和 Shell 脚本并查看结果输出的信息，请参阅 AWS Systems Manager 用户指南 中的 使用 Systems Manager Run Command 运行命令和 Run Command 演练。

示例：使用 Run Command 更新容器实例软件

Run Command 的常用案例是，一次性更新整个容器实例队列中的实例软件。

1. 将 Systems Manager 托管策略附加到 `ecsInstanceRole`。 (p. 193)
2. 在容器实例上安装 SSM 代理。有关更多信息，请参阅 在Amazon ECS-Optimized AMI 上安装 SSM 代理 (p. 193)。
3. 在 https://console.aws.amazon.com/systems-manager 处打开 Systems Manager 控制台。
4. 在左侧导航窗格中，选择 Run Command，然后选择运行命令。
5. 对于 Command document，选择 AWS-RunShellScript。
6. 在 Commands 部分中，输入要发送到容器实例的命令。在本示例中，使用以下命令更新实例软件：

   ```
   $ yum update -y
   ```

7. 在目标实例部分中，选中要运行更新命令的容器实例旁的复选框。
8. 选择 Run 以将命令发送到指定实例。
9. （可选）选择刷新图标以监控命令状态。
10. （可选）在目标和输出中，选择实例 ID 旁边的按钮，然后选择查看输出。

在容器实例启动时启动任务

根据您的应用程序架构设计，您可能需要在每个容器实例上运行一个特定容器以处理操作或安全问题，例如监控、安全性、指标、服务发现或日志记录。

为此，可以将容器实例配置为在启动时或在某些 init 系统（如 Upstart 或 systemd）中使用用户数据脚本调用 docker run 命令。虽然此方法可行，但它有一些缺点，因为 Amazon ECS 不了解容器并且无法监控 CPU、内存、端口或已使用的任何其他资源。要确保 Amazon ECS 可正确了解所有任务资源，请为要在容器实例上运行的容器创建任务定义。然后，使用 Amazon ECS 在启动时利用 Amazon EC2 用户数据放置任务。

以下过程中的 Amazon EC2 用户数据脚本使用 Amazon ECS 自检 API 来确定容器实例。然后，它将使用 AWS CLI 和 start-task 命令在启动期间对自身运行指定任务。

在容器实例启动时启动任务

1. 如果您尚未这样做，请按照创建任务定义 (p. 45)中的过程操作以使用要在启动时在容器实例上运行的容器创建任务定义。

2. 修改您的 `ecsInstanceRole` IAM 角色以便为 `StartTask` API 操作添加权限。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色 (p. 404)。

   a. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在导航窗格中，选择 Roles。
   c. 选择 `ecsInstanceRole`.如果角色不存在，请使用 Amazon ECS 容器实例 IAM 角色 (p. 404)中的过程来创建角色并返回此过程。如果角色存在，请选择角色以查看附加的策略。
   d. 在权限选项卡中，选择 添加内联策略。
   e. 对于服务，依次选择选择服务和 EC2 Container Service。
   f. 对于操作，在搜索字段中键入 StartTask，然后选择 StartTask。
   g. 对于资源，请选择所有资源，然后选择查看策略。
   h. 在查看策略页面上，输入策略的名称，例如 `ecs-start-task`，然后选择创建策略。

3. 通过执行启动 Amazon ECS 容器实例 (p. 165)中的过程（但在Step 6.g (p. 166)中，将下面的 MIME 分段用户数据脚本复制并粘贴到 User data (用户数据) 字段中）使用Amazon ECS-optimized Amazon Linux 2 AMI启动一个或多个容器实例。将 *your_cluster_name* 替换为要注册到的容器实例的集群，并将 *my_task_def* 替换为要在启动时在实例上运行的任务定义。

   Note

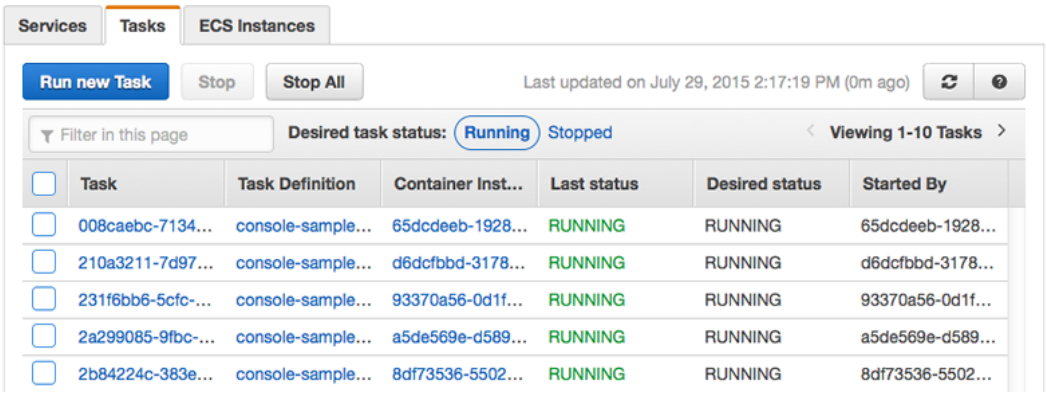
   以下 MIME 分段内容使用 Shell 脚本设置配置值和安装程序包。它还将使用 Upstart 作业在 ecs 服务已在运行且自检 API 可用后启动任务。

```
Content-Type: multipart/mixed; boundary="==BOUNDARY==="
MIME-Version: 1.0

----==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Specify the cluster that the container instance should register into
cluster=your_cluster_name

# Write the cluster configuration variable to the ecs.config file
# (add any other configuration variables here also)
echo ECS_CLUSTER=$cluster >> /etc/ecs/ecs.config

START_TASK_SCRIPT_FILE="/etc/ecs/ecs-start-task.sh"
cat <<- 'EOF' > ${START_TASK_SCRIPT_FILE}
 exec 2>>/var/log/ecs/ecs-start-task.log
 set -x

 # Install prerequisite tools
 yum install -y jq aws-cli

 # Wait for the ECS service to be responsive
 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the container instance ARN and AWS Region from instance metadata
 instance_arn=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $NF}' )
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster' | awk -F/
 '{print $NF}' )
 region=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F: '{print $4}')

 # Specify the task definition to run at launch
 task_definition=my_task_def
```

```
 # Run the AWS CLI start-task command to start your task on this container instance
 aws ecs start-task --cluster $cluster --task-definition $task_definition --container-
instances $instance_arn --started-by $instance_arn --region $region
EOF

# Write systemd unit file
UNIT="ecs-start-task.service"
cat <<- EOF > /etc/systemd/system/${UNIT}
 [Unit]
 Description=ECS Start Task
 Requires=ecs.service
 After=ecs.service

 [Service]
 Restart=always
 ExecStart=/usr/bin/bash ${START_TASK_SCRIPT_FILE}

 [Install]
 WantedBy=default.target
EOF

# Enable our ecs.service dependent service with `--no-block` to prevent systemd
 deadlock
# See https://github.com/aws/amazon-ecs-agent/issues/1707
systemctl enable --now --no-block "${UNIT}"
----BOUNDARY----
```

4. 验证您的容器实例是否启动到正确的集群中以及您的任务是否已启动。

   a. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
   b. 在导航栏中，选择集群所在的区域。
   c. 在导航窗格中，选择 Clusters 并选择托管您的容器实例的集群。
   d. 在 Cluster 页面上，选择 Tasks。



   您启动的每个容器实例均应有在其上运行的任务，并且容器实例 ARN 应在 Started By 列中。

   如果没有看到任务，可以使用 SSH 登录容器实例，在 `/var/log/ecs/ecs-start-task.log` 文件中查看调试信息。

取消注册容器实例

在您使用完容器实例后，可从您的集群中取消注册它。

取消注册后，容器实例再也不能接受新任务。如果在您取消注册容器实例时有正在其上运行的任务，则这些任务将继续运行，直到您终止实例或通过某些其他方法停止任务。

但是，这些任务是孤立的 (不再由 Amazon ECS 监控或说明)。如果您的容器实例上的某个孤立任务是 Amazon ECS 服务的一部分，则该服务计划程序将在其他容器实例上启动此任务的另一个副本 (如果可能)。注册到 传统负载均衡器或应用程序负载均衡器目标组的孤立服务任务中的任何容器均将取消注册。它们将根据负载均衡器或目标组上的设置开始连接耗尽。

如果您打算在容器实例被取消注册后将其用于某个其他用途，则应在取消注册之前停止容器实例上运行的所有任务。这将停止任何孤立任务消耗资源。

> **Important**
>
> 由于每个容器实例均具有唯一状态信息，因此不应将其从一个集群取消注册，然后再将其重新注册到另一个集群。要重新定位容器实例资源，建议您从一个集群终止容器实例，然后在新集群中使用最新的Amazon ECS-optimized Amazon Linux 2 AMI 启动新容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例和启动 Amazon ECS 容器实例 (p. 165)。

取消注册容器实例将从集群中删除该实例，但不会终止 EC2 实例。如果使用完该实例，请务必在 Amazon EC2 控制台中终止它以停止计费。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例。

> **Note**
>
> 如果通过已连接的 Amazon ECS 容器代理终止正在运行的容器实例，则此代理将自动从集群取消注册实例。已停止的容器实例或未连接代理的实例在终止时不会自动取消注册。

取消注册容器实例

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择您的容器实例将注册到的区域。
3. 在导航窗格中，选择 Clusters 并选择托管您的容器实例的集群。
4. 在 Cluster : ***name*** 页面上，选择 ECS Instances。

5. 选择要取消注册的容器实例 ID。
6. 在 Container Instance : ***id*** 页面上，选择 Deregister。
7. 查看取消注册消息，并选择 Yes, Deregister。
8. 如果您已使用完容器实例，则应终止基础 Amazon EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的终止您的实例。

   > **Note**
   >
   > 如果实例是由 Auto Scaling 组或 AWS CloudFormation 堆栈维护的，请通过更新 Auto Scaling 组或 AWS CloudFormation 堆栈终止实例。否则，Auto Scaling 组将在您终止该实例后重新创建它。

Amazon ECS 容器代理

利用 Amazon ECS 容器代理，容器实例可以连接到您的集群。Amazon ECS 容器代理包含在Amazon ECS-optimized AMI 中，但您也可以在支持 Amazon ECS 规格的任何 Amazon EC2 实例上安装它。Amazon ECS 容器代理仅在 Amazon EC2 实例上受支持。

GitHub 上提供了 Amazon ECS 容器代理的源代码。我们鼓励您针对要包含的更改提交提取请求。但是，Amazon Web Services 当前不支持运行此软件的修改后副本。

> **Note**
>
> Amazon ECS 容器代理安装在使用 Fargate 启动类型的任务所使用的 AWS 托管基础设施上。如果您仅使用 Fargate 启动类型的任务，则无需进行额外配置，本主题中的内容不适用。

主题

- 安装 Amazon ECS 容器代理 (p. 198)
- Amazon ECS 容器代理版本 (p. 207)
- 更新 Amazon ECS 容器代理 (p. 211)
- Amazon ECS 容器代理配置 (p. 217)
- 容器实例的私有注册表身份验证 (p. 228)
- 自动化任务和映像清除 (p. 231)
- Amazon ECS 容器元数据文件 (p. 232)
- Amazon ECS 任务元数据终端节点 (p. 236)
- Amazon ECS 容器代理自检 (p. 244)
- HTTP 代理配置 (p. 245)

安装 Amazon ECS 容器代理

如果容器实例不是使用Amazon ECS-optimized AMI 启动的，则可使用以下过程之一手动安装 Amazon ECS 容器代理。

- 对于 Amazon Linux 2 实例，可以使用 `amazon-linux-extras` 命令安装代理。有关更多信息，请参阅在 Amazon Linux 2 EC2 实例上安装 Amazon ECS 容器代理 (p. 198)。
- 对于 Amazon Linux AMI 实例，可以使用 Amazon YUM 存储库安装代理。有关更多信息，请参阅在 Amazon Linux AMI EC2 实例上安装 Amazon ECS 容器代理 (p. 199)。
- 对于非 Amazon Linux 实例，您可以从区域性 S3 存储桶之一或 Docker Hub 下载代理。如果您从区域性 S3 存储桶之一下载，则可以选择使用 PGP 签名来验证容器代理的有效性。有关更多信息，请参阅在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理 (p. 200)

> **Note**
>
> Amazon ECS 容器代理包含在Amazon ECS-optimized AMI 中且不需要安装。

在 Amazon Linux 2 EC2 实例上安装 Amazon ECS 容器代理

要使用 `amazon-linux-extras` 命令在 Amazon Linux 2 EC2 实例上安装 Amazon ECS 容器代理，请使用以下步骤。

在 Amazon Linux 2 EC2 实例上安装 Amazon ECS 容器代理

1. 启动一个 Amazon Linux 2 EC2 实例，该实例具有允许访问 Amazon ECS 的 IAM 角色。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色 (p. 404)。
2. 连接到您的实例。
3. 禁用 `docker` Amazon Linux 额外存储库。`ecs` Amazon Linux 额外存储库附带有自己的 Docker 版本，因此必须禁用 `docker` 额外版本以避免以后出现任何潜在冲突。这可确保您始终使用 Amazon ECS 希望您用于特定版本的容器代理的 Docker 版本。

```
[ec2-user ~]$ sudo amazon-linux-extras disable docker
```

4. 安装并启用 `ecs` Amazon Linux 额外存储库。

```
[ec2-user ~]$ sudo amazon-linux-extras install -y ecs; sudo systemctl enable --now ecs
```

5. (可选) 您可以利用代理自检 API 验证代理是否正在运行并查看有关新容器实例的一些信息。有关更多信息，请参阅the section called “Amazon ECS 容器代理自检” (p. 244)。

```
[ec2-user ~]$ curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

输出:

```
{
    "Cluster": "default",
    "ContainerInstanceArn": "<container_instance_ARN>",
    "Version": "Amazon ECS Agent - v1.29.1 (f95f731b)"
}
```

> **Note**
>
> 如果您未获得响应，请确保在启动实例时已关联 Amazon ECS 容器实例 IAM 角色。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

在 Amazon Linux AMI EC2 实例上安装 Amazon ECS 容器代理

要使用 Amazon YUM 存储库在 Amazon Linux AMI EC2 实例上安装 Amazon ECS 容器代理，请使用以下步骤。

在 Amazon Linux AMI EC2 实例上安装 Amazon ECS 容器代理

1. 启动一个 Amazon Linux AMI EC2 实例，该实例具有允许访问 Amazon ECS 的 IAM 角色。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色 (p. 404)。
2. 连接到您的实例。
3. 安装 `ecs-init` 软件包。有关 `ecs-init` 的更多信息，请参阅 GitHub 上的源代码。

```
[ec2-user ~]$ sudo yum install -y ecs-init
```

4. 启动 Docker 守护程序。

```
[ec2-user ~]$ sudo service docker start
```

输出:

```
Starting cgconfig service:                                 [  OK  ]
Starting docker:                                     [  OK  ]
```

5. 启动 `ecs-init` upstart 作业。

```
[ec2-user ~]$ sudo start ecs
```

输出:

```
ecs start/running, process 2804
```

6. (可选) 您可以利用代理自检 API 验证代理是否正在运行并查看有关新容器实例的一些信息。有关更多信息，请参阅the section called “Amazon ECS 容器代理自检” (p. 244)。

```
[ec2-user ~]$ curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

输出:

```
{
  "Cluster": "default",
  "ContainerInstanceArn": "<container_instance_ARN>",
  "Version": "Amazon ECS Agent - v1.29.1 (f95f731b)"
}
```

在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理

要在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理，您可以从区域性 S3 存储桶之一或 Docker Hub 下载代理。如果您从区域性 S3 存储桶之一下载，则可以选择使用 PGP 签名来验证容器代理的有效性。

下面按区域列出了最新的 Amazon ECS 容器代理文件，以供参考。

Region	Region Name	Container agent	Container agent signature
us-east-2	US East (Ohio)	ECS container agent	PGP signature
us-east-1	US East (N. Virginia)	ECS container agent	PGP signature
us-west-1	US West (N. California)	ECS container agent	PGP signature
us-west-2	US West (Oregon)	ECS container agent	PGP signature
ap-east-1	Asia Pacific (Hong Kong)	ECS container agent	PGP signature
ap-northeast-1	Asia Pacific (Tokyo)	ECS container agent	PGP signature
ap-northeast-2	Asia Pacific (Seoul)	ECS container agent	PGP signature
ap-south-1	Asia Pacific (Mumbai)	ECS container agent	PGP signature
ap-southeast-1	Asia Pacific (Singapore)	ECS container agent	PGP signature

Region	Region Name	Container agent	Container agent signature
ap-southeast-2	Asia Pacific (Sydney)	ECS container agent	PGP signature
ca-central-1	Canada (Central)	ECS container agent	PGP signature
cn-north-1	China (Beijing)	ECS container agent	PGP signature
cn-northwest-1	China (Ningxia)	ECS container agent	PGP signature
eu-central-1	EU (Frankfurt)	ECS container agent	PGP signature
eu-west-1	EU (Ireland)	ECS container agent	PGP signature
eu-west-2	EU (London)	ECS container agent	PGP signature
eu-west-3	EU (Paris)	ECS container agent	PGP signature
sa-east-1	South America (São Paulo)	ECS container agent	PGP signature
us-gov-east-1	AWS GovCloud (US-East)	ECS container agent	PGP signature
us-gov-west-1	AWS GovCloud (US-West)	ECS container agent	PGP signature

在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理

1. 启动一个 Amazon EC2 实例，该实例具有允许访问 Amazon ECS 的 IAM 角色。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。
2. 连接到您的实例。
3. 在实例上安装 Docker。Amazon ECS 需要最低 Docker 版本的 1.9.0 (推荐使用版本 18.06.1-ce)，许多系统程序包管理器中的默认 Docker 版本 (如 yum 或 apt-get) 未满足此最低要求。有关在特定 Linux 分发版上安装最新 Docker 版本的信息，请参阅 https://docs.docker.com/engine/installation/。

   Note

   Amazon Linux AMI 始终包含用于 Amazon ECS 的 Docker 推荐版本。您可以使用 sudo yum install docker -y 命令在 Amazon Linux 上安装 Docker：
4. 检查 Docker 版本以验证系统是否满足最低版本要求。

```
ubuntu:~$ sudo docker version
```

输出:

```
Client version: 1.4.1
Client API version: 1.16
Go version (client): go1.3.3
Git commit (client): 5bc2ff8
OS/Arch (client): linux/amd64
Server version: 1.4.1
Server API version: 1.16
Go version (server): go1.3.3
Git commit (server): 5bc2ff8
```

在本示例中，Docker 版本为 `1.4.1`，低于最低版本 1.9.0。在继续之前，此实例需要升级其 Docker。有关在特定 Linux 分发版上安装最新 Docker 版本的信息，请转到 https://docs.docker.com/engine/installation/。

5. 在容器实例上运行以下命令，以允许端口代理使用回环地址路由流量。

```
ubuntu:~$ sudo sh -c "echo 'net.ipv4.conf.all.route_localnet = 1' >> /etc/sysctl.conf"
ubuntu:~$ sudo sysctl -p /etc/sysctl.conf
```

6. 对容器实例运行以下命令以便为任务启用 IAM 角色。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

```
ubuntu:~$ sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT
 --to-destination 127.0.0.1:51679
ubuntu:~$ sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j
 REDIRECT --to-ports 51679
```

7. 将新的 iptables 配置写入到特定于您的操作系统的位置。

   - 对于 Debian/Ubuntu：

   ```
   sudo sh -c 'iptables-save > /etc/iptables/rules.v4'
   ```

   - 对于 CentOS/RHEL：

   ```
   sudo sh -c 'iptables-save > /etc/sysconfig/iptables'
   ```

8. 创建 `/etc/ecs` 目录并创建 Amazon ECS 容器代理配置文件。

```
ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
```

9. 编辑 `/etc/ecs/ecs.config` 文件并添加以下内容。如果不希望向默认集群注册容器实例，请将 `ECS_CLUSTER` 的值指定为集群名称。

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

有关这些和其他代理运行时选项的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

> **Note**
>
> 您可以选择将代理环境变量存储在 Amazon S3 中（可在启动时使用 Amazon EC2 用户数据将其下载到容器实例）。建议对敏感信息（如私有存储库的身份验证凭证）采用此方法。有关更多信息，请参阅 将容器实例配置存储在 Amazon S3 中 (p. 227) 和 任务的私有注册表身份验证 (p. 104)。

10. 在容器实例上拉取并运行最新的 Amazon ECS 容器代理。

> **Note**
>
> 使用 Docker 重新启动策略或进程管理器（如 upstart 或 systemd）将容器代理作为服务或守护程序处理，并确保在容器代理退出后重新启动它。有关更多信息，请参阅 Docker 文档中的自动启动容器和重新启动策略。Amazon ECS-optimized AMI 的 Linux 变体使用 `ecs-init` RPM

达到此目的，并且您可以在 GitHub 上查看此 RPM 的源代码。有关适用于 Ubuntu 16.04 和 CentOS 7 的示例 systemd 单元文件，请参阅示例容器实例用户数据配置脚本 (p. 171)。

以下示例代理运行命令分为若干个单独的行以显示每个选项。有关这些和其他代理运行时选项的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

Important

运行启用了 SELinux 的操作系统需要在 docker run 命令中使用 `--privileged` 选项。此外，对于启用了 SELinux 的容器实例，建议向 `/log` 和 `/data` 卷挂载添加 `:Z` 选项。但是，在运行该命令前，这些卷的主机挂载必须存在，否则会出现 `no such file or directory` 错误。如果您在启用了 SELinux 的容器实例上运行 Amazon ECS 代理时遇到困难，请执行以下操作：

- 在容器实例上创建主机卷挂载点。

```
ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
```

- 将 `--privileged` 选项添加到下面的 docker run 命令中。
- 将 `:Z` 选项追加到针对下面的 docker run 命令的 `/log` 和 `/data` 容器卷挂载（例如 `--volume=/var/log/ecs/:/log:Z`）。

a. (可选) 从区域性 S3 URL 中下载 ECS 容器代理 tarball 并加载它。如果您未从 S3 下载代理 tarball，则下一步中的 `docker run` 命令会自动从 Docker Hub 中下载。

```
ubuntu:~$ curl -o ecs-agent.tar http://s3.amazonaws.com/amazon-ecs-agent-us-east-1/
ecs-agent-latest.tar
```

Note

要下载其他版本的 Amazon ECS 容器代理，请使用以下格式之一，并更改 URL 中的版本号：

```
ecs-agent-<version>.tar
ecs-agent-<SHA>.tar
```

例如：

```
http://s3.amazonaws.com/amazon-ecs-agent-us-east-1/ecs-agent-v1.18.0.tar
http://s3.amazonaws.com/amazon-ecs-agent-us-east-1/ecs-agent-c0defea9.tar
```

加载 ECS 容器代理映像。

```
ubuntu:~$ sudo docker load --input ./ecs-agent.tar
```

b. 运行 ECS 容器代理映像。

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--net=host \
--env-file=/etc/ecs/ecs.conf \
```

```
amazon/amazon-ecs-agent:latest
```

Important

host 网络模式是容器代理容器的唯一受支持的网络模式。有关更多信息，请参阅使用主机网络模式运行 Amazon ECS 容器代理 (p. 207)。

Note

如果您收到一条 Error response from daemon: Cannot start container 消息，则可以利用 sudo docker rm ecs-agent 命令删除失败的容器并重新尝试运行代理。

11. (可选) 如果您从 S3 下载 Amazon ECS 容器代理文件，则可验证该文件的有效性。

    a. 下载并安装 GnuPG。有关 GnuPG 的更多信息，请参阅 GnuPG 网站。对于 Linux 系统，使用您的 Linux 风格的程序包管理器安装 gpg。

    b. 检索 Amazon ECS PGP 公钥。您可以使用命令检索，也可以手动创建此密钥然后导入。

        i. 选项 1：使用以下命令检索此密钥。

```
gpg --keyserver hkp://keys.gnupg.net --recv BCE9D9A42D51784F
```

        ii. 选项 2：创建包含 Amazon ECS PGP 公钥的以下内容的文件，然后导入此文件：


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
```

8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA

```
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

作为参考的 Amazon ECS PGP 公钥的详细信息：

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

使用以下命令导入 Amazon ECS PGP 公钥。

```
gpg --import <public_key_filename>
```

c. 下载 ECS 容器代理签名。ECS 容器代理签名是存储在扩展名为 `.asc` 的文件中的 ascii 分离 PGP 签名。此签名文件的名称与其对应可执行文件的名称相同，追加了 `.asc`。

```
curl -o ecs-agent.asc http://s3.amazonaws.com/amazon-ecs-agent-us-east-1/ecs-agent-latest.tar.asc
```

d. 验证签名。

```
gpg --verify ecs-agent.asc ./ecs-agent.tar
```

预期输出：

```
gpg: Signature made Wed 16 May 2018 08:21:06 PM UTC using RSA key ID 710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

Note

输出中的警告是预料中的，没有问题；它出现是因为您的个人 PGP 密钥 (如果您有) 和 Amazon ECS PGP 密钥之间没有信任链。有关更多信息，请参阅信任 Web。

使用主机网络模式运行 Amazon ECS 容器代理

运行 Amazon ECS 容器代理时，`ecs-init` 将使用 `host` 网络模式创建容器代理容器。这是容器代理容器的唯一受支持的网络模式。

这使您能够阻止对容器代理启动的容器的 Amazon EC2 实例元数据服务终端节点 (`http://169.254.169.254`) 的访问。这将确保容器无法访问容器实例配置文件中的 IAM 角色凭证并强制任务仅使用 IAM 任务角色凭证。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

这还可以让容器代理不会争用 `docker0` 桥接上的连接和网络流量。

Amazon ECS 容器代理版本

每个 Amazon ECS 容器代理版本都支持不同的功能集并提供了针对早期版本的错误修复。如果可能，我们始终建议使用最新版本的 Amazon ECS 容器代理。要将您的容器代理更新至最新版本，请参阅更新 Amazon ECS 容器代理 (p. 211)。

从最新的Amazon ECS-optimized Amazon Linux 2 AMI 启动容器实例可确保您收到最新的容器代理版本。要使用最新的Amazon ECS-optimized Amazon Linux 2 AMI 启动容器实例，请参阅启动 Amazon ECS 容器实例 (p. 165)。

要在其他操作系统上安装最新版本的 Amazon ECS 容器代理，请参阅安装 Amazon ECS 容器代理 (p. 198)。Amazon ECS-Optimized Amazon Linux 2 AMI 版本 (p. 146)中的表显示了针对每个代理版本在 Amazon Linux 2 上测试过的 Docker 版本。Amazon ECS-Optimized Amazon Linux AMI 容器代理版本 (p. 208)中的表显示了针对每个代理版本在 Amazon Linux AMI 上测试过的 Docker 版本。

要查看每个代理版本包含了哪些功能和增强功能，请参阅 https://github.com/aws/amazon-ecs-agent/releases。

Amazon ECS-Optimized Amazon Linux 2 AMI 容器代理版本

Amazon ECS-optimized Amazon Linux 2 AMI 预先打包有 Amazon ECS 容器代理、Docker 以及用于在启动和关闭时控制代理的启动和停止的 `ecs-init` `systemd` 服务。下表列出了容器代理版本、`ecs-init` 版本以及针对每个Amazon ECS-optimized Amazon Linux 2 AMI 进行了测试和打包的 Docker 版本。

Note

在新的Amazon ECS-optimized Amazon Linux 2 AMI 和 Amazon ECS 代理版本发布时，旧版本仍可在 Amazon EC2 中启动。但是，我们鼓励您更新到最新版本 (p. 211)的 Amazon ECS 代理并使容器实例软件保持最新。如果您通过 AWS Support 请求对旧版本的 Amazon ECS 代理的支持，那么在支持过程中，可能会要求您迁移到最新版本。

Important

Amazon ECS 代理版本 1.20.0 及更高版本已弃用对早于 1.9.0 的 Docker 版本的支持。

Amazon ECS-optimized Amazon Linux 2 AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
20190709	1.29.1	18.06.1-ce	1.29.1-1
20190614	1.29.0	18.06.1-ce	1.29.0-1
20190607	1.29.0	18.06.1-ce	1.29.0-1
20190603	1.28.1	18.06.1-ce	1.28.1-2
20190510	1.28.0	18.06.1-ce	1.28.0-1
20190402	1.27.0	18.06.1-ce	1.27.0-1
20190301	1.26.0	18.06.1-ce	1.26.0-1
20190215	1.25.3	18.06.1-ce	1.25.3-1
20190204	1.25.2	18.06.1-ce	1.25.2-1
20190127	1.25.1	18.06.1-ce	1.25.1-1
20190118	1.25.0	18.06.1-ce	1.25.0-1
20190107	1.24.0	18.06.1-ce	1.24.0-1
20181112	1.22.0	18.06.1-ce	1.22.0-1
20181016	1.20.3	18.06.1-ce	1.21.0-1

有关Amazon ECS-optimized Amazon Linux 2 AMI 的更多信息（包括每个区域中的最新版本的 AMI ID），请参阅Amazon ECS-optimized AMI (p. 127)。

Amazon ECS-Optimized Amazon Linux AMI 容器代理版本

Amazon ECS-optimized Amazon Linux AMI 预先打包有 Amazon ECS 容器代理、Docker 以及用于在启动和关闭时控制代理的启动和停止的 `ecs-init` 服务。下表列出了容器代理版本、`ecs-init` 版本以及针对每个经 Amazon ECS 优化的 AMI 进行了测试和打包的 Docker 版本。

> **Note**
>
> 在新的Amazon ECS-optimized Amazon Linux AMI 和 Amazon ECS 代理版本发布时，旧版本仍可在 Amazon EC2 中启动。但是，我们鼓励您更新到最新版本 (p. 211)的 Amazon ECS 代理并使容器实例软件保持最新。如果您通过 AWS Support 请求对旧版本的 Amazon ECS 代理的支持，那么在支持过程中，可能会要求您迁移到最新版本。

> **Important**
>
> Amazon ECS 代理版本 1.20.0 及更高版本已弃用对早于 1.9.0 的 Docker 版本的支持。

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
2018.03.v	1.29.1	18.06.1-ce	1.29.1-1
2018.03.u	1.29.0	18.06.1-ce	1.29.0-1

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
2018.03.t	`1.29.0`	`18.06.1-ce`	`1.29.0-1`
2018.03.s	`1.28.1`	`18.06.1-ce`	`1.28.1-2`
2018.03.q	`1.28.0`	`18.06.1-ce`	`1.28.0-1`
2018.03.p	`1.27.0`	`18.06.1-ce`	`1.27.0-1`
2018.03.o	`1.26.0`	`18.06.1-ce`	`1.26.0-1`
2018.03.n	`1.25.3`	`18.06.1-ce`	`1.25.3-1`
2018.03.m	`1.25.2`	`18.06.1-ce`	`1.25.2-1`
2018.03.l	`1.25.1`	`18.06.1-ce`	`1.25.1-1`
2018.03.k	`1.25.0`	`18.06.1-ce`	`1.25.0-1`
2018.03.j	`1.24.0`	`18.06.1-ce`	`1.24.0-1`
2018.03.i	`1.22.0`	`18.06.1-ce`	`1.22.0-1`
2018.03.h	`1.21.0`	`18.06.1-ce`	`1.21.0-1`
2018.03.g	`1.20.3`	`18.06.1-ce`	`1.20.3-1`
2018.03.f	`1.20.2`	`18.06.1-ce`	`1.20.2-1`
2018.03.e	`1.20.1`	`18.03.1-ce`	`1.20.1-1`
2018.03.d	`1.20.0`	`18.03.1-ce`	`1.20.0-1`
2018.03.c	`1.19.1`	`18.03.1-ce`	`1.19.1-1`
2018.03.b	`1.19.0`	`18.03.1-ce`	`1.19.0-1`
2018.03.a	`1.18.0`	`17.12.1-ce`	`1.18.0-1`
2017.09.l	`1.17.3`	`17.12.1-ce`	`1.17.3-1`
2017.09.k	`1.17.2`	`17.12.0-ce`	`1.17.2-1`
2017.09.j	`1.17.2`	`17.12.0-ce`	`1.17.2-1`
2017.09.i	`1.17.1`	`17.09.1-ce`	`1.17.1-1`
2017.09.h	`1.17.0`	`17.09.1-ce`	`1.17.0-2`
2017.09.g	`1.16.2`	`17.09.1-ce`	`1.16.2-1`
2017.09.f	`1.16.1`	`17.06.2-ce`	`1.16.1-1`
2017.09.e	`1.16.1`	`17.06.2-ce`	`1.16.1-1`
2017.09.d	`1.16.0`	`17.06.2-ce`	`1.16.0-1`
2017.09.c	`1.15.2`	`17.06.2-ce`	`1.15.1-1`
2017.09.b	`1.15.1`	`17.06.2-ce`	`1.15.1-1`

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**`ecs-init`** version
2017.09.a	`1.15.0`	`17.06.2-ce`	`1.15.0-4`
2017.03.g	`1.14.5`	`17.03.2-ce`	`1.14.5-1`
2017.03.f	`1.14.4`	`17.03.2-ce`	`1.14.4-1`
2017.03.e	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2017.03.d	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2017.03.c	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2017.03.b	`1.14.3`	`17.03.1-ce`	`1.14.3-1`
2016.09.g	`1.14.1`	`1.12.6`	`1.14.1-1`
2016.09.f	`1.14.0`	`1.12.6`	`1.14.0-2`
2016.09.e	`1.14.0`	`1.12.6`	`1.14.0-1`
2016.09.d	`1.13.1`	`1.12.6`	`1.13.1-2`
2016.09.c	`1.13.1`	`1.11.2`	`1.13.1-1`
2016.09.b	`1.13.1`	`1.11.2`	`1.13.1-1`
2016.09.a	`1.13.0`	`1.11.2`	`1.13.0-1`
2016.03.j	`1.13.0`	`1.11.2`	`1.13.0-1`
2016.03.i	`1.12.2`	`1.11.2`	`1.12.2-1`
2016.03.h	`1.12.1`	`1.11.2`	`1.12.1-1`
2016.03.g	`1.12.0`	`1.11.2`	`1.12.0-1`
2016.03.f	`1.11.1`	`1.11.2`	`1.11.1-1`
2016.03.e	`1.11.0`	`1.11.2`	`1.11.0-1`
2016.03.d	`1.10.0`	`1.11.1`	`1.10.0-1`
2016.03.c	`1.10.0`	`1.11.1`	`1.10.0-1`
2016.03.b	`1.9.0`	`1.9.1`	`1.9.0-1`
2016.03.a	`1.8.2`	`1.9.1`	`1.8.2-1`
2015.09.g	`1.8.1`	`1.9.1`	`1.8.1-1`
2015.09.f	`1.8.0`	`1.9.1`	`1.8.0-1`
2015.09.e	`1.7.1`	`1.9.1`	`1.7.1-1`
2015.09.d	`1.7.1`	`1.9.1`	`1.7.1-1`
2015.09.c	`1.7.0`	`1.7.1`	`1.7.0-1`
2015.09.b	`1.6.0`	`1.7.1`	`1.6.0-1`

Amazon ECS-optimized Amazon Linux AMI	Amazon ECS container agent version	Docker version	**ecs-init** version
2015.09.a	1.5.0	1.7.1	1.5.0-1
2015.03.g	1.4.0	1.7.1	1.4.0-2
2015.03.f	1.4.0	1.6.2	1.4.0-1
2015.03.e	1.3.1	1.6.2	1.3.1-1
2015.03.d	1.2.1	1.6.2	1.2.0-2
2015.03.c	1.2.0	1.6.2	1.2.0-1
2015.03.b	1.1.0	1.6.0	1.0-3
2015.03.a	1.0.0	1.5.0	1.0-1

有关Amazon ECS-optimized Amazon Linux AMI 的更多信息（包括每个区域中的最新版本的 AMI ID），请参阅Amazon ECS-optimized AMI (p. 127)。

更新 Amazon ECS 容器代理

有时，您可能需要更新 Amazon ECS 容器代理以获取错误修正和新功能。更新 Amazon ECS 容器代理不会中断容器实例上正在运行的任务或服务。更新代理的过程各不相同，具体取决于您的容器实例是使用 Amazon ECS-optimized AMI 还是其他操作系统启动的。

> **Note**
>
> 代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

主题

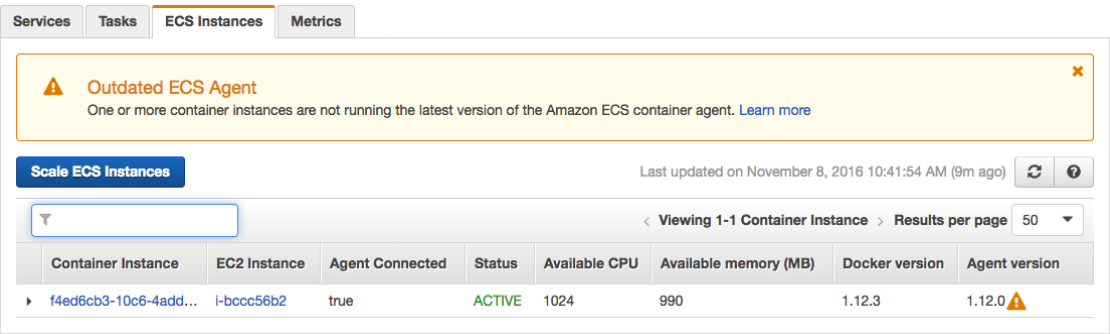
- 检查 Amazon ECS 容器代理版本 (p. 211)
- 更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理 (p. 213)
- 手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI） (p. 215)

检查 Amazon ECS 容器代理版本

您可以查看在容器实例上运行的容器代理的版本以确定是否需要更新它。Amazon ECS 控制台中的容器实例视图提供了代理版本。使用以下过程可以检查代理版本。

检查 Amazon ECS 容器代理是否在最新版本的控制台中运行

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters 页面上，选择托管容器实例或要检查的实例的集群。
3. 在 Cluster : ***cluster_name*** 页面上，选择 ECS Instances。
4. 注意容器实例的 Agent version 列。如果容器实例未包含最新版本的容器代理，则控制台将通过消息来提醒您并标记已过期的代理版本。



如果您的代理版本是 1.29.1，则表明您运行的是最新的容器代理。如果您的代理版本低于 1.29.1，则可以使用以下过程更新容器代理：

- 如果您的容器实例正在运行Amazon ECS-optimized AMI，请参阅更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理 (p. 213)。
- 如果您的容器实例没有运行Amazon ECS-optimized AMI，请参阅手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI）(p. 215)。

> **Important**
>
> 要在经 Amazon ECS 优化的 AMI 上从 v1.0.0 之前的版本更新 Amazon ECS 代理版本，建议您终止当前容器实例并启动具有最新 AMI 的新实例。使用预览版的任何容器实例都应停用并更换为最新的 AMI。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。

您还可以使用 Amazon ECS 容器代理自检 API 从容器实例本身检查代理版本。有关更多信息，请参阅 Amazon ECS 容器代理自检 (p. 244)。

利用自检 API 检查 Amazon ECS 容器代理运行的是否为最新版本

1. 通过 SSH 登录到容器实例。
2. 查询自检 API。

```
[ec2-user ~]$ curl -s 127.0.0.1:51678/v1/metadata | python -mjson.tool
```

输出:

```
{
    "Cluster": "default",
    "ContainerInstanceArn": "arn:aws:ecs:us-west-2:<aws_account_id>:container-
instance/4d3910c1-27c8-410c-b1df-f5d06fab4305",
    "Version": "Amazon ECS Agent - v1.29.1 (f95f731b)"
}
```

> **Note**
>
> 自检 API 在 v1.0.0 版本的 Amazon ECS 容器代理中添加了 `Version` 信息。如果在查询自检 API 时 `Version` 不存在，或者自检 API 在您的代理中根本不存在，则表明运行的版本是 v0.0.3 或更早的版本。您应该更新版本。

更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理

如果您使用Amazon ECS-optimized AMI，则可通过多种方式获取最新版本的 Amazon ECS 容器代理（按建议的顺序显示）：

- 终止您当前的容器实例并启动最新版本的Amazon ECS-optimized Amazon Linux 2 AMI（手动执行，或者通过使用最新的 AMI 更新 Auto Scaling 启动配置来执行）。这将提供全新的容器实例，其中包含最新的经测试和验证的版本的 Amazon Linux、Docker、`ecs-init` 和 Amazon ECS 容器代理。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。
- 使用 SSH 连接到实例，并将 `ecs-init` 程序包（及其依赖项）更新到最新版本。此操作提供了最新的经测试和验证的版本的 Docker 和 `ecs-init`（它们在 Amazon Linux 存储库中提供）以及最新版本的 Amazon ECS 容器代理。有关更多信息，请参阅更新Amazon ECS-optimized AMI 上的 `ecs-init` 程序包 (p. 213)。
- 使用 `UpdateContainerAgent` API 操作更新容器代理（通过控制台执行，或者通过 AWS CLI 或 AWS 开发工具包执行）。有关更多信息，请参阅使用 `UpdateContainerAgent` API 操作更新 Amazon ECS 容器代理 (p. 213)。

> **Note**
>
> 代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

更新Amazon ECS-optimized AMI 上的 **`ecs-init`** 程序包

1. 通过 SSH 登录到容器实例。有关更多信息，请参阅 连接到您的容器实例 (p. 180)。
2. 使用以下命令更新 `ecs-init` 包。

```
[ec2-user ~]$ sudo yum update -y ecs-init
```

> **Note**
>
> 将立即更新 `ecs-init` 程序包和 Amazon ECS 容器代理。但是，更新版本的 Docker 直至 Docker 守护程序重新启动后才会加载。通过重启实例或通过在您的实例上运行以下命令来重新启动：
>
> - Amazon ECS-optimized Amazon Linux 2 AMI：
>
> ```
> sudo systemctl restart docker
> ```
>
> - Amazon ECS-optimized Amazon Linux AMI：
>
> ```
> sudo service docker restart && sudo start ecs
> ```

使用 `UpdateContainerAgent` API 操作更新 Amazon ECS 容器代理

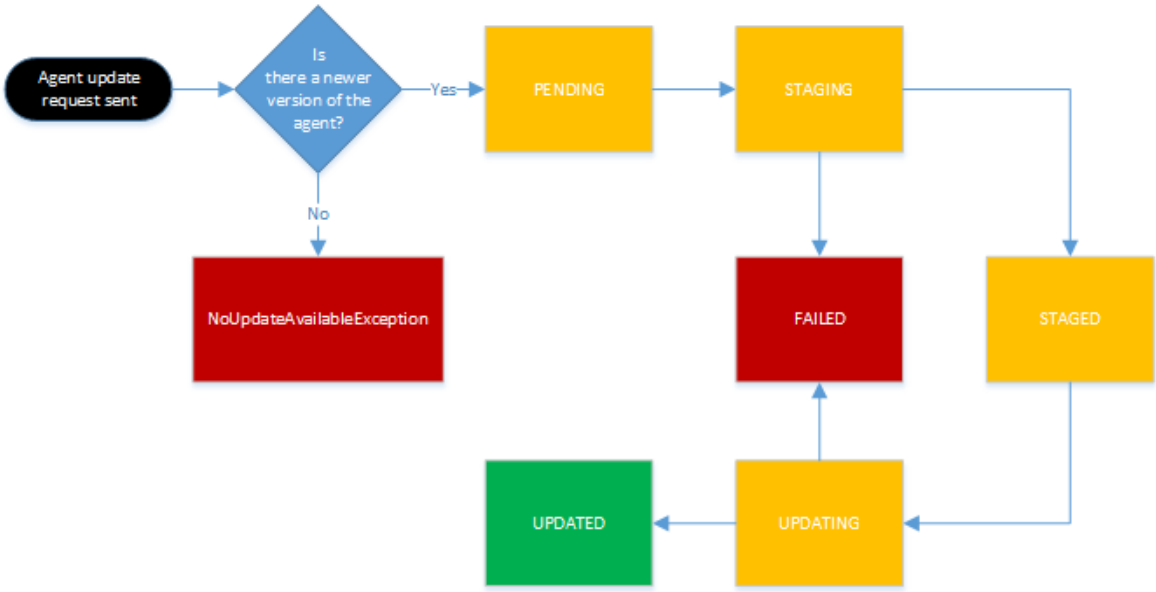
> **Important**
>
> 此更新过程仅在Amazon ECS-optimized AMI 的 Linux 变体上受支持。有关运行其他操作系统的容器实例，请参阅手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI） (p. 215)。

Note

使用 `UpdateContainerAgent` API 操作的代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

要在Amazon ECS-optimized AMI 上从 v1.0.0 之前的版本更新 Amazon ECS 代理版本，建议您终止当前容器实例并使用最新的 AMI 版本启动新实例。使用预览版的任何容器实例都应停用并更换为最新的 AMI。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。

更新过程在您请求代理更新时开始（通过控制台执行，或者通过 AWS CLI 或 AWS 开发工具包执行）。Amazon ECS 将对照最新发布的代理版本检查您当前的代理版本，如果有可用的更新，则更新过程将按以下流程图所示的方式进行。如果更新不可用（例如代理已在运行最新版本），则会返回 `NoUpdateAvailableException`。



上面显示的更新过程的各个阶段如下所示：

`PENDING`

代理更新可用，并且更新过程已开始。

`STAGING`

代理已开始下载代理更新。如果代理无法下载更新，或者更新的内容不正确或已损坏，代理将发送失败通知，并且更新将变为 `FAILED` 状态。

`STAGED`

代理下载已完成，且代理内容已经过验证。

`UPDATING`

`ecs-init` 服务已重新启动，并且选取了新代理版本。如果代理出于某个原因无法重新启动，更新将变为 `FAILED` 状态；否则，代理将向 Amazon ECS 发送更新已完成的信号。

在控制台中更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理

Note

代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters 页面上，选择托管容器实例或要检查的实例的集群。
3. 在 Cluster : ***cluster_name*** 页面上，选择 ECS Instances。
4. 选择要更新的容器实例。
5. 在 Container Instance 页面上，选择 Update agent。

使用 AWS CLI 更新Amazon ECS-optimized AMI 上的 Amazon ECS 容器代理

Note

使用 `UpdateContainerAgent` API 操作的代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

- 使用以下命令更新容器实例上的 Amazon ECS 容器代理：

```
aws ecs update-container-agent --cluster cluster_name --container-
instance container_instance_id
```

手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI）

手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-optimized AMI）

Note

代理更新不适用于 Windows 容器实例。我们建议您启动新的容器实例来更新您的 Windows 集群中的代理版本。

1. 通过 SSH 登录到容器实例。
2. 检查您的代理是否使用 `ECS_DATADIR` 环境变量保存其状态。

```
ubuntu:~$ docker inspect ecs-agent | grep ECS_DATADIR
```

输出:

```
"ECS_DATADIR=/data",
```

Important

如果上一个命令未返回 `ECS_DATADIR` 环境变量，则您在更新代理前必须停止在此容器实例上运行的任何任务。具有 `ECS_DATADIR` 环境变量的较新的代理将保存其状态，您可以在任务运行的同时更新它们，而不会出现问题。

3. 停止 Amazon ECS 容器代理。

```
ubuntu:~$ docker stop ecs-agent
```

4. 删除代理容器。

```
ubuntu:~$ docker rm ecs-agent
```

5. 确保 `/etc/ecs/ecs.config` 中存在 `/etc/ecs` 目录和 Amazon ECS 容器代理配置文件。

```
ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
```

6. 编辑 `/etc/ecs/ecs.config` 文件，并确保它至少包含以下变量声明。如果不希望向默认集群注册容器实例，请将 `ECS_CLUSTER` 的值指定为集群名称。

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

有关这些和其他代理运行时选项的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

> **Note**
>
> 您可以选择将代理环境变量存储在 Amazon S3 中（可在启动时使用 Amazon EC2 用户数据将其下载到容器实例）。建议对敏感信息（如私有存储库的身份验证凭证）采用此方法。有关更多信息，请参阅 将容器实例配置存储在 Amazon S3 中 (p. 227) 和 任务的私有注册表身份验证 (p. 104)。

7. 从 Docker Hub 中拉取最新的 Amazon ECS 容器代理映像。

```
ubuntu:~$ docker pull amazon/amazon-ecs-agent:latest
```

输出:

```
Pulling repository amazon/amazon-ecs-agent
a5a56a5e13dc: Download complete
511136ea3c5a: Download complete
9950b5d678a1: Download complete
c48ddcf21b63: Download complete
Status: Image is up to date for amazon/amazon-ecs-agent:latest
```

8. 在容器实例上运行最新的 Amazon ECS 容器代理。

> **Note**
>
> 使用 Docker 重新启动策略或进程管理器（如 upstart 或 systemd）将容器代理作为服务或守护程序处理，并确保在容器代理退出后重新启动它。有关更多信息，请参阅 Docker 文档中的自动启动容器和重新启动策略。Amazon ECS-optimized AMI 使用 `ecs-init` RPM 达到此目的，并且您可以在 GitHub 上查看此 RPM 的源代码。有关适用于 Ubuntu 16.04 和 CentOS 7 的示例 systemd 单元文件，请参阅示例容器实例用户数据配置脚本 (p. 171)。

以下示例代理运行命令分为若干个单独的行以显示每个选项。有关这些和其他代理运行时选项的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

> **Important**
>
> 运行启用了 SELinux 的操作系统需要在 docker run 命令中使用 `--privileged` 选项。此外，对于启用了 SELinux 的容器实例，建议向 `/log` 和 `/data` 卷挂载添加 `:z` 选项。但是，在运行该命令前，这些卷的主机挂载必须存在，否则会出现 `no such file or directory` 错误。如果您在启用了 SELinux 的容器实例上运行 Amazon ECS 代理时遇到困难，请执行以下操作：
>
> - 在容器实例上创建主机卷挂载点。

```
ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
```

- 将 `--privileged` 选项添加到下面的 docker run 命令中。
- 将 `:Z` 选项追加到针对下面的 docker run 命令的 `/log` 和 `/data` 容器卷挂载（例如 `--volume=/var/log/ecs/:/log:Z`）。

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
```

Note

如果您收到一条 `Error response from daemon: Cannot start container` 消息，则可以利用 sudo docker rm ecs-agent 命令删除失败的容器并重新尝试运行代理。

Amazon ECS 容器代理配置

Amazon ECS 容器代理支持很多部署选项，其中大多数选项应通过环境变量来设置。提供了以下环境变量，并且它们都是可选的。

如果容器实例是使用Amazon ECS-optimized AMI 的 Linux 变体启动的，则可以在 `/etc/ecs/ecs.config` 文件中设置这些环境变量，然后重新启动代理。您还可以在启动时将这些配置变量写入到具有 Amazon EC2 用户数据的容器实例。有关更多信息，请参阅 使用 Amazon EC2 用户数据引导启动容器实例 (p. 168)。

如果要手动启动 Amazon ECS 容器代理（对于非Amazon ECS-optimized AMI），则可以在用于启动代理的 docker run 命令中使用这些环境变量（使用语法 `--env=VARIABLE_NAME=VARIABLE_VALUE`）。对于敏感信息 (如私有存储库的身份验证凭证)，您应该将代理环境变量存储在一个文件中并利用 `--env-file path_to_env_file` 选项一次性将它们传递完。

主题

- 可用参数 (p. 217)
- 将容器实例配置存储在 Amazon S3 中 (p. 227)

可用参数

下面是可用的环境密钥：

`ECS_CLUSTER`

示例值：`MyCluster`

Linux 上的默认值：`default`

Windows 上的默认值：`default`

此代理应签入的集群。如果此值未定义，则假定使用 `default` 集群。如果 `default` 集群不存在，Amazon ECS 容器代理会尝试创建它。如果指定了非 `default` 集群，但该集群不存在，注册将失败。

`ECS_RESERVED_PORTS`

示例值：`[22, 80, 5000, 8080]`

Linux 上的默认值：`[22, 2375, 2376, 51678, 51679, 51680]`

Windows 上的默认值：`[53, 135, 139, 445, 2375, 2376, 3389, 5985, 51678, 51679]`

应标记为不可用于在此容器实例上的计划的一组端口。

`ECS_RESERVED_PORTS_UDP`

示例值：`[53, 123]`

Linux 上的默认值：`[]`

Windows 上的默认值：`[]`

应标记为不可用于在此容器实例上的计划的一组 UDP 端口。

`ECS_ENGINE_AUTH_TYPE`

示例值：`dockercfg | docker`

Linux 上的默认值：Null

Windows 上的默认值：Null

对于私有注册表身份验证是必需的。这是 `ECS_ENGINE_AUTH_DATA` 中的身份验证数据的类型。有关更多信息，请参阅 身份验证格式 (p. 228)。

`ECS_ENGINE_AUTH_DATA`

示例值：

- `ECS_ENGINE_AUTH_TYPE=dockercfg`：`{"https://index.docker.io/v1/": {"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}`
- `ECS_ENGINE_AUTH_TYPE=docker`：`{"https://index.docker.io/v1/": {"username":"my_name","password":"my_password","email":"email@example.com"}}`

Linux 上的默认值：Null

Windows 上的默认值：Null

对于私有注册表身份验证是必需的。如果 `ECS_ENGINE_AUTH_TYPE=dockercfg`，则 `ECS_ENGINE_AUTH_DATA` 值应该是通过运行 docker login 创建的 Docker 配置文件（`~/.dockercfg` 或 `~/.docker/config.json`）的内容。如果 `ECS_ENGINE_AUTH_TYPE=docker`，则 `ECS_ENGINE_AUTH_DATA` 值应该是身份验证所针对的注册表服务器的 JSON 表示形式，以及注册表所需的身份验证参数（如该账户的用户名、密码和电子邮件地址）。有关更多信息，请参阅身份验证格式 (p. 228)。

`AWS_DEFAULT_REGION`

示例值：`us-east-1`

Linux 上的默认值：从 EC2 实例元数据中获取。

Windows 上的默认值：从 EC2 实例元数据中获取。

要在 API 请求中使用以及用于推断正确的后端主机的区域。

`AWS_ACCESS_KEY_ID`

示例值：`AKIAIOSFODNN7EXAMPLE`

Linux 上的默认值：从 EC2 实例元数据中获取。

Windows 上的默认值：从 EC2 实例元数据中获取。

由代理用于所有调用的访问密钥。

`AWS_SECRET_ACCESS_KEY`

示例值：`wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

Linux 上的默认值：从 EC2 实例元数据中获取。

Windows 上的默认值：从 EC2 实例元数据中获取。

由代理用于所有调用的私有密钥。

`AWS_SESSION_TOKEN`

Linux 上的默认值：从 EC2 实例元数据中获取。

Windows 上的默认值：从 EC2 实例元数据中获取。

用于临时凭证的会话令牌。

`DOCKER_HOST`

示例值：`unix:///var/run/docker.sock`

Linux 上的默认值：`unix:///var/run/docker.sock`

Windows 上的默认值：`npipe:////./pipe/docker_engine`

用于创建与 Docker 守护程序的连接；行为方式与 Docker 客户端使用的环境变量相似。

`ECS_LOGLEVEL`

示例值：`crit | error | warn | info | debug`

Linux 上的默认值：`info`

Windows 上的默认值：`info`

要在 `stdout` 上进行日志记录的级别。

`ECS_LOGFILE`

示例值：`/ecs-agent.log`

Linux 上的默认值：Null

Windows 上的默认值：Null

变量指定的路径位于容器内。如果您通过 `ecs-init` 运行代理（这是使用经过 Amazon ECS 优化的 AMI 时的默认方法），则容器内路径将为 `/log`，并且 `ecs-init` 会将其装载到主机上的 `/var/log/ecs/`。

日志记录级别由 `ECS_LOGLEVEL` 控制。如果为空白，则不会记录任何日志。如果设置此值，则将调试级别 (无论 `ECS_LOGLEVEL` 如何) 的日志写入到该文件。

`ECS_CHECKPOINT`

示例值：`true | false`

Linux 上的默认值：如果 `ECS_DATADIR` 显式设置为非空值，则 `ECS_CHECKPOINT` 将设置为 `true`；否则，它将设置为 `false`。

Windows 上的默认值：如果 `ECS_DATADIR` 显式设置为非空值，则 `ECS_CHECKPOINT` 将设置为 `true`；否则，它将设置为 `false`。

是否要将检查点状态保存到 `ECS_DATADIR` 指定的位置。

`ECS_DATADIR`

示例值：`/data`

Linux 上的默认值：`/data/`

Windows 上的默认值：`C:\ProgramData\Amazon\ECS\data`

正在运行 Amazon ECS 容器代理的容器上的永久数据目录的名称。该目录用于保存有关集群和代理状态的信息。

`ECS_UPDATES_ENABLED`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

在请求 ECS 代理更新时是否退出以进行更新。

`ECS_UPDATE_DOWNLOAD_DIR`

示例值：`/cache`

Linux 上的默认值：Null

Windows 上的默认值：Null

要放置容器中的已下载的更新 tarball 的文件系统位置。

`ECS_DISABLE_METRICS`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`true`

是否对 Amazon ECS 禁用 CloudWatch 指标。如果此值设置为 `true`，则不会收集 CloudWatch 指标。

`ECS_POLL_METRICS`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

是否在收集任务的 CloudWatch 指标时进行轮询或流式传输。

`ECS_POLLING_METRICS_WAIT_DURATION`

示例值：`30s`

Linux 上的默认值：`15s`

Windows 上的默认值：`15s`

等待轮询任务的新 CloudWatch 指标的时间。仅在 `ECS_POLL_METRICS` 为 `true` 时使用。

`ECS_RESERVED_MEMORY`

示例值：32

Linux 上的默认值：0

Windows 上的默认值：0

要从分配给您的任务的池中删除的内存量 (MiB)。这将为关键系统进程 (包括 Docker 守护程序和 Amazon ECS 容器代理) 有效预留内存。例如，如果您指定 `ECS_RESERVED_MEMORY=256`，则代理将为该实例注册总内存减去 256 MiB 后得到的内存量，256 MiB 的系统内存无法由 ECS 任务分配。有关更多信息，请参阅容器实例内存管理 (p. 190)。

`ECS_AVAILABLE_LOGGING_DRIVERS`

示例值：`["awslogs","fluentd","gelf","json-file","journald","splunk","logentries","syslog"]`

Linux 上的默认值：`["json-file","none"]`

Windows 上的默认值：`["json-file","none"]`

> **Note**
>
> 如果您使用的是 ECS init，默认值为 `["json-file","syslog","awslogs","none"]`。

容器实例上可用的日志记录驱动程序。在容器实例上运行的 Amazon ECS 容器代理必须先将该实例上可用的日志记录驱动程序注册到 `ECS_AVAILABLE_LOGGING_DRIVERS` 环境变量，然后放置在该实例上的容器才能在任务中使用这些驱动程序的日志配置选项。有关如何使用 `awslogs` 日志驱动程序的信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。有关可用于 Docker 版本的不同日志驱动程序的更多信息，请参阅 Docker 文档中的配置日志记录驱动程序。

`ECS_DISABLE_PRIVILEGED`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

是否在容器实例上禁用启动特权容器。如果此值设置为 `true`，则不允许使用特权容器。

`ECS_SELINUX_CAPABLE`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

SELinux 是否在容器实例上可用。

`ECS_APPARMOR_CAPABLE`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

AppArmor 是否在容器实例上可用。

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

示例值：`1h` (有效时间单位为“ns”、“us”(或“µs”)、“ms”、“s”、“m”和“h”。)

Linux 上的默认值：`3h`

Windows 上的默认值：`3h`

从任务停止到删除 Docker 容器所等待的持续时间。由于这将删除 Docker 容器数据，因此请注意，如果此值设置得过低，您可能无法检查已停止的容器或无法在日志删除前查看它们。最短持续时间为 `1m`；任何短于 1 分钟的值都会被忽略。

`ECS_CONTAINER_STOP_TIMEOUT`

示例值：`10m` (有效时间单位为“ns”、“us”(或“µs”)、“ms”、“s”、“m”和“h”。)

Linux 上的默认值：`30s`

Windows 上的默认值：`30s`

从任务停止到其容器由于未自行正常退出而被强制终止所等待的持续时间。

`ECS_CONTAINER_START_TIMEOUT`

示例值：`10m` (有效时间单位为“ns”、“us”(或“µs”)、“ms”、“s”、“m”和“h”。)

Linux 上的默认值：`3m`

Windows 上的默认值：`8m`

在放弃启动容器之前等待的时间期限。

`HTTP_PROXY`

示例值：`10.0.0.131:3128`

Linux 上的默认值：Null

Windows 上的默认值：Null

供 ECS 代理用来连接到 Internet 的 HTTP 代理的主机名 (或 IP 地址)和端口号 (例如，在您的容器实例无法通过 Amazon VPC Internet 网关或 NAT 网关/实例的访问外部网络的情况下)。如果设置了此变量，您还必须设置 `NO_PROXY` 变量以筛选来自该代理的 EC2 实例元数据和 Docker 守护程序流量。有关更多信息，请参阅 HTTP 代理配置 (p. 245)。

`NO_PROXY`

示例值：

- Linux: `169.254.169.254,169.254.170.2,/var/run/docker.sock`
- Windows: `169.254.169.254,169.254.170.2,\\.\pipe\docker_engine`

Linux 上的默认值：Null

Windows 上的默认值：Null

不应转发到指定的 `HTTP_PROXY` 的 HTTP 流量。您必须指定 `169.254.169.254,/var/run/docker.sock` 以筛选来自该代理的 EC2 实例元数据和 Docker 守护程序流量。有关更多信息，请参阅HTTP 代理配置 (p. 245)。

`ECS_ENABLE_TASK_IAM_ROLE`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

> **Note**
>
> 如果您使用的是 ECS init，默认值为 `true`。

是否应在具有 `bridge` 或 `default` 网络模式的任务容器的容器实例上启用任务的 IAM 角色。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

> **Note**
>
> 如果您使用的是 ECS init，默认值为 `true`。

是否应在具有 `host` 网络模式的任务容器的容器实例上启用任务的 IAM 角色。此变量仅在代理版本 1.12.0 和更高版本上受支持。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

`ECS_DISABLE_IMAGE_CLEANUP`

示例值：`true`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

是否为 Amazon ECS 代理禁用自动化映像清除。有关更多信息，请参阅 自动化任务和映像清除 (p. 231)。

`ECS_IMAGE_CLEANUP_INTERVAL`

示例值：`30m`

Linux 上的默认值：`30m`

Windows 上的默认值：`30m`

自动化映像清除周期之间的时间间隔。如果设置为短于 10 分钟，该值将被忽略。

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

示例值：`30m`

Linux 上的默认值：`1h`

Windows 上的默认值：`1h`

拉取映像与可考虑将其进行自动化映像清除之间的最短时间间隔。

`ECS_NUM_IMAGES_DELETE_PER_CYCLE`

示例值：`5`

Linux 上的默认值：`5`

Windows 上的默认值：`5`

要在单次自动化映像清除周期中删除的映像的最大数量。如果设置为小于 1，该值将被忽略。

`ECS_IMAGE_PULL_BEHAVIOR`

示例值：`default | always | once | prefer-cached`

Linux 上的默认值：`default`

Windows 上的默认值：`default`

用于为您的容器实例自定义拉取映像流程的行为。下面介绍了可选行为：

- 如果指定 `default`，则远程拉取映像。如果映像拉取失败，容器将在实例上使用缓存的映像。
- 如果指定 `always`，则始终远程拉取映像。如果映像拉取失败，任务将失败。此选项可确保始终拉取映像的最新版本。所有缓存映像均被忽略，并受自动映像清除过程支配。
- 如果指定 `once`，仅当同一容器实例上之前的任务尚未拉取映像或缓存映像已被自动映像清除过程删除时远程拉取映像。否则，将使用实例上缓存的映像。这样可确保不会进行不必要的映像拉取尝试。
- 如果指定 `prefer-cached`，则在没有缓存映像时远程拉取映像。否则，将使用实例上缓存的映像。为容器禁用自动映像清除，以确保不会删除缓存的映像。

`ECS_IMAGE_PULL_INACTIVITY_TIMEOUT`

示例值：`1m`

Linux 上的默认值：`1m`

Windows 上的默认值：`3m`

docker 提取完成后等待提取容器的时间。用于优化大型 Windows 容器。

`ECS_INSTANCE_ATTRIBUTES`

示例值：`{"custom_attribute": "custom_attribute_value"}`

Linux 上的默认值：Null

Windows 上的默认值：Null

自定义属性列表为 JSON 形式，应用于您的容器实例。在实例注册时使用该属性会添加自定义属性，这样可跳过通过 AWS 管理控制台 手动添加自定义属性的方法。

> **Note**
>
> 添加的属性将不会应用于已注册的容器实例。要向已注册的容器实例添加自定义属性，请参阅添加属性 (p. 258)。

有关可用自定义属性的信息，请参阅属性 (p. 257)。

此变量的无效 JSON 值导致代理退出，代码为 `5`。代理日志中会显示一条消息。JSON 值可能有效，但验证属性时发现问题（例如，值过长或包含无效字符）。在这种情况下，仍会进行容器实例注册，但代理会退出，代码为 `5`，代理日志中也将写入一条消息。有关如何查找代理日志的信息，请参阅Amazon ECS 容器代理日志 (p. 589)。

`ECS_ENABLE_TASK_ENI`

示例值：`true` | `false`

Linux 上的默认值：`false`

Windows 上的默认值：不适用

是否为要使用其自己的网络接口启动的任务启用任务联网。

`ECS_CNI_PLUGINS_PATH`

示例值：`/ecs/cni`

Linux 上的默认值：`/amazon-ecs-cni-plugins`

Windows 上的默认值：不适用

cni 二进制文件所在的路径。

`ECS_AWSVPC_BLOCK_IMDS`

示例值：`true` | `false`

Linux 上的默认值：`false`

Windows 上的默认值：不适用

是否阻止使用 `awsvpc` 网络模式启动的任务对实例元数据的访问。

`ECS_AWSVPC_ADDITIONAL_LOCAL_ROUTES`

示例值：`["10.0.15.0/24"]`

Linux 上的默认值：`[]`

Windows 上的默认值：不适用

在 `awsvpc` 网络模式中，将通过主桥而不是任务弹性网络接口路由至这些前缀的流量。

`ECS_ENABLE_CONTAINER_METADATA`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

当为 `true` 时，代理会创建一个文件，用于描述容器的元数据。此文件可以通过使用容器环境变量 `$ECS_CONTAINER_METADATA_FILE` 来定位和使用。

`ECS_HOST_DATA_DIR`

示例值：`/var/lib/ecs`

Linux 上的默认值：`/var/lib/ecs`

Windows 上的默认值：不适用

安装 `ECS_DATADIR` 的主机上的源目录。在 ECS 代理作为容器运行的情况下，我们使用它来确定容器元数据文件的源装载路径。我们不在 Windows 中使用此值，因为 ECS 代理并不作为容器运行。

`ECS_ENABLE_TASK_CPU_MEM_LIMIT`

示例值：`true | false`

Linux 上的默认值：`true`

Windows 上的默认值：`false`

是否启用任务级 CPU 和内存限制。

`ECS_CGROUP_PATH`

示例值：`/sys/fs/cgroup`

Linux 上的默认值：`/sys/fs/cgroup`

Windows 上的默认值：不适用

ECS 代理需要的根 cgroup 路径。这是可从代理装载访问的路径。

`ECS_ENABLE_CPU_UNBOUNDED_WINDOWS_WORKAROUND`

示例值：`true | false`

Linux 上的默认值：不适用

Windows 上的默认值：`false`

当为 `true` 时，ECS 将允许 CPU 未绑定 (CPU=0) 任务与 CPU 绑定任务一起在 Windows 中运行。

`ECS_TASK_METADATA_RPS_LIMIT`

示例值：`100,150`

Linux 上的默认值：`40,60`

Windows 上的默认值：`40,60`

任务元数据终端节点的稳定状态和突增阈值限制的整数值 (用逗号分隔)。

`ECS_SHARED_VOLUME_MATCH_FULL_CONFIG`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

当任务定义中指定了 `dockerVolumeConfiguration` 且使用了 `autoprovision` 标志时，Amazon ECS 容器代理将比较 Docker 卷详细信息和现有 Docker 卷的详细信息。当 `ECS_SHARED_VOLUME_MATCH_FULL_CONFIG` 为 `true` 时，容器代理对卷的完整配置（`name`、`driverOpts` 和 `labels`）进行比较以验证卷是否相同。当它为 `false` 时，容器代理将使用 Docker 的默认行为，仅验证该卷的 `name`。如果某个卷在容器实例之间共享，则应设置为 `false`。有关更多信息，请参阅 Docker 卷 (p. 85)。

`ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM`

示例值：`ec2_instance`

Linux 上的默认值：无

Windows 上的默认值：无

如果指定了 `ec2_instance`，则容器实例上定义的现有标签将注册到 Amazon ECS。可使用 `ListTagsForResource` 操作发现标签。与容器实例关联的 IAM 角色应具有允许的 `ec2:DescribeTags` 操作。有关更多信息，请参阅 向容器实例添加标签 (p. 333)。

`ECS_CONTAINER_INSTANCE_TAGS`

示例值：`{"tag_key": "tag_val"}`

Linux 上的默认值：`{}`

Windows 上的默认值：`{}`

应用到容器实例的元数据，可帮助您对资源进行分类和组织。每个标签都包含自定义键和可选值。标签键最多可包含 128 个字符。标签值最多可包含 256 个字符。

如果容器实例标签是使用 `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` 参数传播的，则这些标签将由使用 `ECS_CONTAINER_INSTANCE_TAGS` 指定的标签覆盖。有关更多信息，请参阅向容器实例添加标签 (p. 333)。

`ECS_ENABLE_UNTRACKED_IMAGE_CLEANUP`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

是否允许 ECS 代理删除不属于 Amazon ECS 任务的容器和镜像。

`ECS_EXCLUDE_UNTRACKED_IMAGE`

示例值：`{"alpine":"latest"}`

Linux 上的默认值：`{}`

Windows 上的默认值：`{}`

镜像的逗号分隔列表 (`imageName:tag`)，如果 `ECS_ENABLE_UNTRACKED_IMAGE_CLEANUP` 为 `true`，则 ECS 代理不应删除该列表。

`ECS_DISABLE_DOCKER_HEALTH_CHECK`

示例值：`true | false`

Linux 上的默认值：`false`

Windows 上的默认值：`false`

是否为 ECS 代理禁用 Docker 容器运行状况检查。

`ECS_NVIDIA_RUNTIME`

示例值：`nvidia`

Linux 上的默认值：`nvidia`

Windows 上的默认值：`n/a`

要用于将 NVIDIA GPU 设备传递到容器的运行时。如果已指定 GPU 资源要求，则不应将此参数指定为任务定义中的环境变量。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。

将容器实例配置存储在 Amazon S3 中

Amazon ECS 容器代理配置通过上面所述的环境变量来控制。Amazon ECS-optimized AMI 的 Linux 变体将在容器代理启动时在 `/etc/ecs/ecs.config` 中查找这些变量，并相应地配置代理。某些无害环境变量（如 `ECS_CLUSTER`）可在启动时通过 Amazon EC2 用户数据传递到容器实例，并可写入到此文件而不会造成任何后果。但是，其他敏感信息（如您的 AWS 凭证或 `ECS_ENGINE_AUTH_DATA` 变量）不应通过用户数据传递到某个实例或以它们将在 `.bash_history` 文件中显示的方式写入到 `/etc/ecs/ecs.config`。

将配置信息存储在 Amazon S3 中的私有存储桶中并向您的容器实例 IAM 角色授予只读访问权限，这是一个允许在启动时配置容器实例的安全方便的方法。您可以将 `ecs.config` 文件的副本存储在私有存储桶中，再使用 Amazon EC2 用户数据安装 AWS CLI，然后在实例启动时将配置信息复制到 `/etc/ecs/ecs.config`。

为容器实例角色授予 Amazon S3 只读访问权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选择要用于您的容器实例的 IAM 角色（此角色可能名为 `ecsInstanceRole`）。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。
3. 在 Managed Policies 下，选择 Attach Policy。
4. 要缩小策略结果的范围，请在附加策略页面上，为筛选条件键入 `S3`。
5. 选中 AmazonS3ReadOnlyAccess 策略左侧的框并选择 Attach Policy。

在 Amazon S3 中存储 **`ecs.config`** 文件

1. 采用以下格式，利用`ecs.config`中的有效环境变量和值创建Amazon ECS 容器代理配置 (p. 217)文件。此示例将配置私有注册表身份验证。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。

```
ECS_ENGINE_AUTH_TYPE=dockercfg
```

```
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

2. 要存储您的配置文件，请在 Amazon S3 中创建私有存储桶。有关更多信息，请参阅 Amazon Simple Storage Service 入门指南 中的创建存储桶。
3. 将 `ecs.config` 文件上传到 S3 存储桶。有关更多信息，请参阅 Amazon Simple Storage Service 入门指南 中的将对象添加到存储桶。

在启动时从 Amazon S3 加载 **`ecs.config`** 文件

1. 完成本节中的上述过程以允许 Amazon S3 对您的容器实例进行只读访问，并将 `ecs.config` 文件存储在私有 S3 存储桶中。
2. 通过执行启动 Amazon ECS 容器实例 (p. 165)中的步骤启动新的容器实例。在Step 6.g (p. 166)中，使用以下示例脚本。该脚本将安装 AWS CLI 并将您的配置文件复制到 `/etc/ecs/ecs.config`。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

容器实例的私有注册表身份验证

Amazon ECS 容器代理可使用基本身份验证对私有注册表（包括 Docker Hub）进行身份验证。当启用私有注册表身份验证时，您可以在任务定义中使用私有 Docker 映像。只有使用 EC2 启动类型的任务才支持此功能。

另一种启用私有注册表身份验证的方法是使用 AWS Secrets Manager 在容器定义中安全地存储并随后引用您的私有注册表凭证。这样，您的任务就可以使用私有存储库中的映像。此方法支持使用 EC2 或 Fargate 启动类型的任务。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。

Amazon ECS 容器代理在启动时将查找两个环境变量：

- `ECS_ENGINE_AUTH_TYPE`，指定要发送的身份验证数据的类型。
- `ECS_ENGINE_AUTH_DATA`，包含实际身份验证凭证。

当容器实例启动时以及每次启动服务（利用 sudo start ecs 命令）时，Amazon ECS-optimized AMI 的 Linux 变体将在 `/etc/ecs/ecs.config` 文件中扫描这些变量。未经 Amazon ECS 优化的 AMI 应将这些环境变量存储在一个文件中，然后利用 `--env-file` *`path_to_env_file`* 选项将其传递到用于启动容器代理的 docker run 命令。

> **Important**
>
> 我们不建议在实例启动时利用 Amazon EC2 用户数据注入这些身份验证环境变量或利用 `--env` 选项将其传递到 docker run 命令。这些方法不适合敏感数据，如身份验证凭证。有关将身份验证凭证安全地添加到容器实例的信息，请参阅将容器实例配置存储在 Amazon S3 中 (p. 227)。

身份验证格式

私有注册表身份验证有两种可用格式：`dockercfg` 和 `docker`。

dockercfg 身份验证格式

`dockercfg` 格式使用存储在您在运行 docker login 命令时创建的配置文件中的身份验证信息。您可以通过在本地系统上运行 docker login 并输入注册表用户名、密码和电子邮件地址来创建此文件。您也可以登录到容器实例并运行命令。根据您的 Docker 版本的不同，此文件将以 `~/.dockercfg` 或 `~/.docker/config.json` 的形式保存。

```
cat ~/.docker/config.json
```

输出:

```
{
  "auths": {
    "https://index.docker.io/v1/": {
      "auth": "zq212MzEXAMPLE7o6T25Dk0i"
    }
  }
}
```

> **Important**
>
> 较新版本的 Docker 可使用外部 `auths` 对象创建上面所示的配置文件。Amazon ECS 代理仅支持以下格式的 `dockercfg` 身份验证数据，无需 `auths` 对象。如果已安装 jq 实用程序，则可以使用以下命令提取此数据：cat ~/.docker/config.json | jq .auths

```
cat ~/.docker/config.json | jq .auths
```

输出:

```
{
  "https://index.docker.io/v1/": {
    "auth": "zq212MzEXAMPLE7o6T25Dk0i",
    "email": "email@example.com"
  }
}
```

在上面的示例中，以下环境变量应添加到 Amazon ECS 容器代理在运行时加载的环境变量文件中（对于Amazon ECS-optimized AMI，该文件为 `/etc/ecs/ecs.config`）。如果没有使用Amazon ECS-optimized AMI 并且要利用 docker run 手动启动代理，请在启动代理时利用 `--env-file` *`path_to_env_file`* 选项指定环境变量文件。

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

您可以使用以下语法配置多个私有注册表：

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example-01.com"},"repo.example-02.com":
{"auth":"fQ172MzEXAMPLEoF7225DU0j","email":"email@example-02.com"}}
```

docker 身份验证格式

`docker` 格式使用注册表服务器（代理应向其进行身份验证）的 JSON 表示形式。它还包括注册表需要的身份验证参数（如该账户的用户名、密码和电子邮件地址）。对于 Docker Hub 账户，JSON 表示形式如下所示：

```
{
  "https://index.docker.io/v1/": {
    "username": "my_name",
    "password": "my_password",
    "email": "email@example.com"
  }
```

```
}
```

在此示例中，以下环境变量应添加到 Amazon ECS 容器代理在运行时加载的环境变量文件中（对于Amazon ECS-optimized AMI，该文件为 `/etc/ecs/ecs.config`）。如果没有使用Amazon ECS-optimized AMI 并且要利用 docker run 手动启动代理，请在启动代理时利用 `--env-file` *`path_to_env_file`* 选项指定环境变量文件。

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

您可以使用以下语法配置多个私有注册表：

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"username":"my_name","password":"my_password","email":"email@example-01.com"},"repo.example-02.com":
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

启用私有注册表

使用以下过程可为容器实例启用私有注册表。

在Amazon ECS-optimized AMI 中启用私有注册表

1. 使用 SSH 登录到您的容器实例。
2. 打开 `/etc/ecs/ecs.config` 文件并为注册表和账户添加 `ECS_ENGINE_AUTH_TYPE` 和 `ECS_ENGINE_AUTH_DATA` 值：

   ```
   sudo vi /etc/ecs/ecs.config
   ```

   此示例将对 Docker Hub 用户账户进行身份验证：

   ```
   ECS_ENGINE_AUTH_TYPE=docker
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"username":"my_name","password":"my_password","email":"email@example.com"}}
   ```

3. 检查您的代理是否使用 `ECS_DATADIR` 环境变量保存其状态：

   ```
   docker inspect ecs-agent | grep ECS_DATADIR
   ```

   输出:

   ```
   "ECS_DATADIR=/data",
   ```

   > **Important**
   >
   > 如果上一个命令未返回 `ECS_DATADIR` 环境变量，则您在停止代理前必须停止在此容器实例上运行的任何任务。使用 `ECS_DATADIR` 环境变量的较新的代理将会保存其状态，您可以在任务运行的过程中停止和启动它们，而不会造成问题。有关更多信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

4. 停止 `ecs` 服务：

   ```
   sudo stop ecs
   ```

输出:

```
ecs stop/waiting
```

5. 重新启动 ecs 服务。

   - 对于Amazon ECS-optimized Amazon Linux 2 AMI：

     ```
     sudo systemctl restart ecs
     ```

   - 对于Amazon ECS-optimized Amazon Linux AMI：

     ```
     sudo stop ecs && sudo start ecs
     ```

6. (可选) 您可以通过查询代理自检 API 操作，验证代理是否正在运行并查看有关新容器实例的一些信息。有关更多信息，请参阅 the section called “Amazon ECS 容器代理自检” (p. 244)。

   ```
   curl http://localhost:51678/v1/metadata
   ```

   输出:

   ```
   {
     "Cluster": "default",
     "ContainerInstanceArn": "<container_instance_ARN>",
     "Version": "Amazon ECS Agent - v1.29.1 (f95f731b)"
   }
   ```

自动化任务和映像清除

每次将任务放在一个容器实例上时，Amazon ECS 容器代理都会检查该任务中引用的映像是否为存储库中的指定标记的最新版本。如果不是，默认行为将允许代理从各个存储库中拉取映像。如果频繁更新任务和服务中的映像，则容器实例存储可能很快会被没在使用并且可能再也不会使用的 Docker 映像填满。例如，如果您使用了连续集成和连续部署 (CI/CD) 管道。

> **Note**
>
> 可以使用 `ECS_IMAGE_PULL_BEHAVIOR` 参数自定义 Amazon ECS 代理映像拉取行为。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。

同样，属于已停止任务的容器也可能会通过日志信息、数据卷和其他项目消耗容器实例存储空间。这些项目对于调试意外停止的容器很有用，但此存储的大部分空间在一段时间后可以安全地释放。

默认情况下，Amazon ECS 容器代理将自动清除已停止的任务和您的容器实例上的任何任务均未在使用的 Docker 映像。

> **Note**
>
> 自动映像清除功能要求 Amazon ECS 容器代理的版本最低为 1.13.0。要将您的代理更新至最新版本，请参阅更新 Amazon ECS 容器代理 (p. 211)。

可调节参数

以下代理配置变量可用于调整自动化任务和映像清除体验。有关如何在容器实例上设置这些变量的更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

此变量指定在删除属于已停止任务的任何容器前的等待时间。只要存在引用某个映像的容器，映像清除过程就无法删除该映像。当映像不再由任何容器（已停止或正在运行）引用后，则会变成清除的候选项。默认情况下，此参数设置为 3 小时，但如果您的应用程序有需要，您最多可以将此时间间缩短为 1 分钟。

`ECS_DISABLE_IMAGE_CLEANUP`

如果您将此变量设置为 `true`，则会在您的容器实例上禁用自动映像清除，且不会自动删除映像。

`ECS_IMAGE_CLEANUP_INTERVAL`

此变量指定自动映像清除过程应检查要删除的映像的频率。默认值为每 30 分钟一次，但您可以将此时间缩短至 10 分钟，以便更频繁地删除映像。

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

此变量指定拉取映像与映像可能变成删除的候选项之间的最短时间。这可用于防止清除刚刚拉取的映像。默认值为 1 小时。

`ECS_NUM_IMAGES_DELETE_PER_CYCLE`

此变量指定在一个清除周期内可以清除的映像数。默认值为 5，最小值为 1。

清除工作流程

当 Amazon ECS 容器代理正在运行且自动化映像清除未禁用时，该代理将按照 `ECS_IMAGE_CLEANUP_INTERVAL` 变量确定的频率，检查是否有未被正在运行或已停止的容器引用的 Docker 映像。如果发现未使用的映像，并且这些映像的存在时间超过了 `ECS_IMAGE_MINIMUM_CLEANUP_AGE` 指定的最短清除时间，则该代理将删除由 `ECS_NUM_IMAGES_DELETE_PER_CYCLE` 变量指定的最大数量的映像。最早引用的映像会最先删除。在删除映像后，该代理将等到下一个时间间隔并重复上述流程。

Amazon ECS 容器元数据文件

自 Amazon ECS 容器代理的 1.15.0 版本起，ECS 任务容器中提供了各种容器元数据。通过启用此功能，您可以查询有关任务、容器以及容器内容器实例的信息。您也可以通过读取每个容器的元数据文件来从容器实例中获取该信息。元数据文件在主机实例上创建，并作为 Docker 卷挂载到容器中。

容器元数据文件位置设置为容器内的 `ECS_CONTAINER_METADATA_FILE` 环境变量。您可以使用以下命令读取容器中的文件内容：

- 对于 Linux 实例：

```
cat $ECS_CONTAINER_METADATA_FILE
```

- 对于 Windows 实例 (PowerShell)：

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

清除容器时，主机实例上的容器元数据文件将被清除。在这种情况下，您可以使用 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 容器代理变量进行调整。有关更多信息，请参阅 自动化任务和映像清除 (p. 231)。

主题

- 启用容器元数据 (p. 233)

- 容器元数据文件位置 (p. 233)
- 容器元数据文件格式 (p. 234)

启用容器元数据

该功能在默认情况下处于禁用状态。您可以将 `ECS_ENABLE_CONTAINER_METADATA` 容器代理变量设置为 `true`，从而在容器实例级别启用容器元数据。您可以在 `/etc/ecs/ecs.config` 配置文件中设置此变量，并重新启动代理。您也可以在启动代理容器时将其设置为 Docker 环境变量。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

> **Note**
>
> 支持此功能的最低 Amazon ECS 容器代理版本是 1.15.0。

如果在代理启动时将 `ECS_ENABLE_CONTAINER_METADATA` 设置为 `true`，则会为 ECS 启动的任何将来容器创建元数据文件。

> **Note**
>
> Amazon ECS 容器代理无法为那些在 `ECS_ENABLE_CONTAINER_METADATA` 容器代理变量设置为 `true` 之前创建的容器创建元数据文件。为确保所有容器都收到元数据文件，应在容器实例启动时设置此代理变量。

容器元数据文件位置

默认情况下，容器元数据文件写入到以下主机和容器路径。

- 对于 Linux 实例：
  - 主机路径：`/var/lib/ecs/data/metadata/`*`task_id`*`/`*`container_name`*`/ecs-container-metadata.json`

    > **Note**
    >
    > Linux 主机路径假定在启动代理时使用默认数据目录挂载路径 (`/var/lib/ecs/data`)。如果您并非使用Amazon ECS-optimized AMI（或 `ecs-init` 程序包来启动和维护容器代理），请确保将 `ECS_HOST_DATA_DIR` 代理配置变量设置为容器代理状态文件所在的主机路径。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。
  - 容器路径：`/opt/ecs/metadata/`*`random_ID`*`/ecs-container-metadata.json`
- 对于 Windows 实例：
  - 主机路径：`C:\ProgramData\Amazon\ECS\data\metadata\`*`task_id`*`\`*`container_name`*`\ecs-container-metadata.json`
  - 容器路径：`C:\ProgramData\Amazon\ECS\metadata\`*`random_ID`*`\ecs-container-metadata.json`

但是为了便于访问，容器元数据文件位置设置为容器内的 `ECS_CONTAINER_METADATA_FILE` 环境变量。您可以使用以下命令读取容器中的文件内容：

- 对于 Linux 实例：

```
cat $ECS_CONTAINER_METADATA_FILE
```

- 对于 Windows 实例 (PowerShell)：

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

容器元数据文件格式

以下信息存储在容器元数据 JSON 文件中。

`Cluster`

容器的任务在其上运行的集群的名称。

`ContainerInstanceARN`

主机容器实例的完整 Amazon 资源名称 (ARN)。

`TaskARN`

容器所属的任务的完整 Amazon 资源名称 (ARN)。

`ContainerID`

容器的 Docker 容器 ID (而不是 Amazon ECS 容器 ID)。

`ContainerName`

容器 Amazon ECS 任务定义中的容器名称。

`DockerContainerName`

Docker 守护程序用于容器的容器名称（例如，docker ps 命令输出中显示的名称）。

`ImageID`

用于启动容器的 Docker 映像的 SHA 摘要。

`ImageName`

用于启动容器的 Docker 映像的映像名称和标签。

`PortMappings`

与容器关联的任何端口映射。

- `ContainerPort`

  公开的容器上的端口。

- `HostPort`

  公开的主机容器实例上的端口。

- `BindIp`

  Docker 分配给容器的绑定 IP 地址。此 IP 地址仅适用于 `bridge` 网络模式，只能从容器实例进行访问。

- `Protocol`

  用于端口映射的网络协议。

`Networks`

容器的网络模式和 IP 地址。

- `NetworkMode`

  容器所属的任务的网络模式。

- `IPv4Addresses`

  与容器关联的 IP 地址。

> **Important**
>
> 如果您的任务使用的是 `awsvpc` 网络模式，则将不会返回容器的 IP 地址。在这种情况下，您可以通过使用以下命令读取 /etc/hosts 文件来检索 IP 地址：
>
> ```
> cat /etc/hosts | tail -1 | awk {'print $1'}
> ```

`MetadataFileStatus`

元数据文件的状态。如果状态为 `READY`，则元数据文件是最新状态且内容完整。如果文件尚未准备就绪 (例如在启动任务时)，将会提供文件格式的截断版本。为了避免可能出现的竞用情况，比如已经启动容器，但尚未写入元数据，您可以解析元数据文件，并等待此参数设置为 `READY`，然后再使用元数据。这通常是在容器启动后的 1 秒内提供。

Example Amazon ECS 容器元数据文件 (**READY**)

以下示例显示了处于 `READY` 状态的容器元数据文件。

```
{
 "Cluster": "default",
 "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/1f73d099-
b914-411c-a9ff-81633b7741dd",
 "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/2b88376d-aba3-4950-9ddf-bcb0f388a40c",
 "ContainerID": "98e44444008169587b826b4cd76c6732e5899747e753af1e19a35db64f9e9c32",
 "ContainerName": "metadata",
 "DockerContainerName": "/ecs-metadata-7-metadata-f0edfbd6d09fdef20800",
 "ImageID": "sha256:c24f66af34b4d76558f7743109e2476b6325fcf6cc167c6e1e07cd121a22b341",
 "ImageName": "httpd:2.4",
 "PortMappings": [
  {
   "ContainerPort": 80,
   "HostPort": 80,
   "BindIp": "",
   "Protocol": "tcp"
  }
 ],
 "Networks": [
  {
   "NetworkMode": "bridge",
   "IPv4Addresses": [
    "172.17.0.2"
   ]
  }
 ],
 "MetadataFileStatus": "READY"
}
```

Example 未完成的 Amazon ECS 容器元数据文件（尚未处于 **READY** 状态）

以下示例显示了尚未处于 `READY` 状态的容器元数据文件。文件中的信息仅限任务定义中已知的几个参数。容器元数据文件应在容器启动后的 1 秒内准备就绪。

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-
instance/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
```

```
}
```

Amazon ECS 任务元数据终端节点

Amazon ECS 容器代理提供了检索各种任务元数据和 Docker 统计数据的方法。这称为任务元数据终端节点。提供了以下版本：

- 任务元数据终端节点版本 3 – 适用于在平台版本 v1.3.0 或更高版本上使用 Fargate 启动类型的任务，以及使用 EC2 启动类型并在最低运行 1.21.0 版的 Amazon ECS 容器代理的 Amazon EC2 基础设施上启动的任务。有关更多信息，请参阅 任务元数据终端节点版本 3 (p. 236)。
- 任务元数据终端节点版本 2 – 适用于在平台版本 v1.1.0 或更高版本上使用 Fargate 启动类型的任务以及使用 EC2 启动类型、还使用 `awsvpc` 网络模式并在最低运行 1.17.0 版的 Amazon ECS 容器代理的 Amazon EC2 基础设施上启动的任务。有关更多信息，请参阅任务元数据终端节点版本 2 (p. 240)。

有关查询元数据和统计信息 API 终端节点的示例 Go 应用程序的信息，请参阅https://github.com/aws/amazon-ecs-agent/blob/2bf4348a0ff89e23be4e82a6c5ff28edf777092c/misc/taskmetadata-validator/taskmetadata-validator.go。

主题

- 任务元数据终端节点版本 3 (p. 236)
- 任务元数据终端节点版本 2 (p. 240)

任务元数据终端节点版本 3

从 Amazon ECS 容器代理版本 1.21.0 开始，代理将称为 `ECS_CONTAINER_METADATA_URI` 的环境变量注入任务中的每个容器。在您查询任务元数据版本 3 终端节点时，将为任务提供各种任务元数据和 Docker 统计数据。

启用任务元数据

任务元数据终端节点版本 3 功能默认为在平台版本 v1.3.0 或更高版本上使用 Fargate 启动类型的任务以及使用 EC2 启动类型并在最低运行 1.21.0 版的 Amazon ECS 容器代理的 Amazon EC2 基础设施上启动的任务启用。有关更多信息，请参阅Amazon ECS 容器代理版本 (p. 207)。

您可以通过将代理更新为最新版本来增加在旧容器实例上对于该功能的支持。有关更多信息，请参阅 更新 Amazon ECS 容器代理 (p. 211)。

> **Important**
>
> 对于使用 Fargate 启动类型和 v1.3.0 之前的平台版本的任务，支持任务元数据版本 2 终端节点。有关更多信息，请参阅任务元数据终端节点版本 2 (p. 240)。

任务元数据终端节点版本 3 路径

以下任务元数据终端节点可用于容器：

`${ECS_CONTAINER_METADATA_URI}`

此路径返回容器的元数据 JSON。

`${ECS_CONTAINER_METADATA_URI}/task`

此路径返回任务的元数据 JSON，包括与任务相关的所有容器的 ID 和名称列表。有关此终端节点响应的更多信息，请参阅任务元数据 JSON 响应 (p. 237)。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路径返回特定 Docker 容器的 Docker 统计数据 JSON。有关每个返回的统计信息的更多信息，请参阅 Docker API 文档中的 ContainerStats。

`${ECS_CONTAINER_METADATA_URI}/task/stats`

此路径返回与任务相关的所有容器的 Docker 统计数据 JSON。有关每个返回的统计信息的更多信息，请参阅 Docker API 文档中的 ContainerStats。

任务元数据 JSON 响应

以下信息返回自任务元数据终端节点 (`${ECS_CONTAINER_METADATA_URI}/task`) JSON 响应。

`Cluster`

The Amazon ECS cluster to which the task belongs.

`TaskARN`

The full Amazon Resource Name (ARN) of the task to which the container belongs.

`Family`

The family of the Amazon ECS task definition for the task.

`Revision`

The revision of the Amazon ECS task definition for the task.

`DesiredStatus`

The desired status for the task from Amazon ECS.

`KnownStatus`

The known status for the task from Amazon ECS.

`Containers`

A list of container metadata for each container associated with the task.

- `DockerId`

  The Docker ID for the container.

- `Name`

  The name of the container as specified in the task definition.

- `DockerName`

  The name of the container supplied to Docker. The Amazon ECS container agent generates a unique name for the container to avoid name collisions when multiple copies of the same task definition are run on a single instance.

- `Image`

  The image for the container.

- `ImageID`

  The SHA-256 digest for the image.

- `Ports`

  Any ports exposed for the container. This parameter is omitted if there are no exposed ports.

- `Labels`

  Any labels applied to the container. This parameter is omitted if there are no labels applied.

`DesiredStatus`

The desired status for the container from Amazon ECS.

`KnownStatus`

The known status for the container from Amazon ECS.

`ExitCode`

The exit code for the container. This parameter is omitted if the container has not exited.

`Limits`

The resource limits specified at the container level (such as CPU and memory). This parameter is omitted if no resource limits are defined.

`CreatedAt`

The time stamp for when the container was created. This parameter is omitted if the container has not been created yet.

`StartedAt`

The time stamp for when the container started. This parameter is omitted if the container has not started yet.

`FinishedAt`

The time stamp for when the container stopped. This parameter is omitted if the container has not stopped yet.

`Type`

The type of the container. Containers that are specified in your task definition are of type `NORMAL`. You can ignore other container types, which are used for internal task resource provisioning by the Amazon ECS container agent.

`Networks`

The network information for the container, such as the network mode and IP address. This parameter is omitted if no network information is defined.

`Limits`

The resource limits specified at the task level (such as CPU and memory). This parameter is omitted if no resource limits are defined.

`PullStartedAt`

The time stamp for when the first container image pull began.

`PullStoppedAt`

The time stamp for when the last container image pull finished.

`ExecutionStoppedAt`

The time stamp for when the tasks `DesiredStatus` moved to `STOPPED`. This occurs when an essential container moves to `STOPPED`.

`AvailabilityZone`

The Availability Zone the task is in.

任务元数据响应示例

以下是有关单容器任务的 JSON 响应。

```
{
```

```
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID": "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
```

```
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

任务元数据终端节点版本 2

从 1.17.0 版 Amazon ECS 容器代理开始，各种任务元数据和 Docker 统计数据都可用于那些在 Amazon ECS 容器代理所提供的 HTTP 终端节点上使用 `awsvpc` 网络模式的任务。

所有属于使用 `awsvpc` 网络模式启动的任务的容器都会收到预定义本地链路地址范围内的一个本地 IPv4 地址。当一个容器查询元数据终端节点时，Amazon ECS 容器代理会基于该容器的唯一 IP 地址确定容器属于哪个任务，并返回有关该任务的元数据和统计数据。

启用任务元数据

为以下任务默认启用任务元数据版本 2 功能：

- 使用 Fargate 启动类型并使用平台版本 v1.1.0 或更高版本的任务。有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。
- 使用 EC2 启动类型、还使用 `awsvpc` 网络模式并在最低运行 1.17.0 版的 Amazon ECS 容器代理的 Amazon EC2 基础设施上启动的任务。有关更多信息，请参阅Amazon ECS 容器代理版本 (p. 207)。

您可以通过将代理更新为最新版本来增加在旧容器实例上对于该功能的支持。有关更多信息，请参阅 更新 Amazon ECS 容器代理 (p. 211)。

任务元数据终端节点路径

以下 API 终端节点可用于容器：

`169.254.170.2/v2/metadata`

此终端节点返回任务的元数据 JSON，包括与任务相关的所有容器的 ID 和名称列表。有关此终端节点响应的更多信息，请参阅任务元数据 JSON 响应 (p. 240)。

`169.254.170.2/v2/metadata/<container-id>`

此终端节点为指定 Docker 容器 ID 返回元数据 JSON。

`169.254.170.2/v2/stats`

此终端节点为所有与此任务相关的容器返回 Docker 统计信息 JSON。有关每个返回的统计信息的更多信息，请参阅 Docker API 文档中的 ContainerStats。

`169.254.170.2/v2/stats/<container-id>`

此终端节点为指定 Docker 容器 ID 返回 Docker 统计数据 JSON。有关每个返回的统计信息的更多信息，请参阅 Docker API 文档中的 ContainerStats。

任务元数据 JSON 响应

以下信息返回自任务元数据终端节点 (`169.254.170.2/v2/metadata`) JSON 响应。

`Cluster`

The Amazon ECS cluster to which the task belongs.

`TaskARN`

The full Amazon Resource Name (ARN) of the task to which the container belongs.

`Family`

The family of the Amazon ECS task definition for the task.

`Revision`

The revision of the Amazon ECS task definition for the task.

`DesiredStatus`

The desired status for the task from Amazon ECS.

`KnownStatus`

The known status for the task from Amazon ECS.

`Containers`

A list of container metadata for each container associated with the task.

`DockerId`

The Docker ID for the container.

`Name`

The name of the container as specified in the task definition.

`DockerName`

The name of the container supplied to Docker. The Amazon ECS container agent generates a unique name for the container to avoid name collisions when multiple copies of the same task definition are run on a single instance.

`Image`

The image for the container.

`ImageID`

The SHA-256 digest for the image.

`Ports`

Any ports exposed for the container. This parameter is omitted if there are no exposed ports.

`Labels`

Any labels applied to the container. This parameter is omitted if there are no labels applied.

`DesiredStatus`

The desired status for the container from Amazon ECS.

`KnownStatus`

The known status for the container from Amazon ECS.

`ExitCode`

The exit code for the container. This parameter is omitted if the container has not exited.

`Limits`

The resource limits specified at the container level (such as CPU and memory). This parameter is omitted if no resource limits are defined.

`CreatedAt`

The time stamp for when the container was created. This parameter is omitted if the container has not been created yet.

`StartedAt`

The time stamp for when the container started. This parameter is omitted if the container has not started yet.

`FinishedAt`

The time stamp for when the container stopped. This parameter is omitted if the container has not stopped yet.

`Type`

The type of the container. Containers that are specified in your task definition are of type `NORMAL`. You can ignore other container types, which are used for internal task resource provisioning by the Amazon ECS container agent.

`Networks`

The network information for the container, such as the network mode and IP address. This parameter is omitted if no network information is defined.

`Limits`

The resource limits specified at the task level (such as CPU and memory). This parameter is omitted if no resource limits are defined.

`PullStartedAt`

The time stamp for when the first container image pull began.

`PullStoppedAt`

The time stamp for when the last container image pull finished.

`ExecutionStoppedAt`

The time stamp for when the tasks `DesiredStatus` moved to `STOPPED`. This occurs when an essential container moves to `STOPPED`.

`AvailabilityZone`

The Availability Zone the task is in.

任务元数据响应示例

以下是有关单容器任务的 JSON 响应。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
```

```
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID": "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
```

```
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 容器代理自检

Amazon ECS 容器代理提供了一个 API 操作，用于收集有关正在运行该代理的容器实例以及在该实例上正在运行的相关任务的详细信息。您可以在容器实例内使用 curl 命令查询 Amazon ECS 容器代理（端口 51678）并返回容器实例元数据或任务信息。

> **Important**
>
> 您的容器实例必须具有相应的 IAM 角色，以允许访问 Amazon ECS 来检索元数据。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

要查看容器实例元数据，请通过 SSH 登录到容器实例并运行以下命令。元数据包含容器实例 ID、在其中注册容器实例的 Amazon ECS 集群和 Amazon ECS 容器代理版本信息。

```
[ec2-user ~]$ curl -s http://localhost:51678/v1/metadata | python -mjson.tool
```

输出:

```
{
  "Cluster": "default",
  "ContainerInstanceArn": "<container_instance_ARN>",
  "Version": "Amazon ECS Agent - v1.29.1 (f95f731b)"
}
```

要查看有关在容器实例上运行的所有任务的信息，请通过 SSH 登录到容器实例并运行以下命令：

```
[ec2-user ~]$ curl http://localhost:51678/v1/tasks
```

输出:

```
{
  "Tasks": [
    {
      "Arn": "arn:aws:ecs:us-east-1:<aws_account_id>:task/example5-58ff-46c9-
ae05-543f8example",
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Family": "hello_world",
      "Version": "8",
      "Containers": [
        {
          "DockerId": "9581a69a761a557fbfce1d0f6745e4af5b9dbfb86b6b2c5c4df156f1a5932ff1",
          "DockerName": "ecs-hello_world-8-mysql-fcae8ac8f9f1d89d8301",
          "Name": "mysql"
        },
        {
          "DockerId": "bf25c5c5b2d4dba68846c7236e75b6915e1e778d31611e3c6a06831e39814a15",
          "DockerName": "ecs-hello_world-8-wordpress-e8bfddf9b488dff36c00",
          "Name": "wordpress"
        }
      ]
```

```
    }
  ]
}
```

您可以查看在容器实例上运行的特定任务的信息。要指定特定的任务或容器，请向请求中追加以下内容之一：

- 任务 ARN (`?taskarn=task_arn`)
- 容器的 Docker ID (`?dockerid=docker_id`)

要获取有关容器 Docker ID 的任务信息，请通过 SSH 登录到容器实例并运行以下命令。

> **Note**
>
> 1.14.2 之前的 Amazon ECS 容器代理版本要求为自检 API 提供完整的 Docker 容器 ID，而不是利用 docker ps 显示的简略版。您可以通过在容器实例上运行 docker ps --no-trunc 命令来获取容器的完整 Docker ID。

```
[ec2-user ~]$ curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

输出:

```
{
    "Arn": "arn:aws:ecs:us-east-1:<aws_account_id>:task/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId": "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx"
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

HTTP 代理配置

您可以将 Amazon ECS 容器实例配置为对 Amazon ECS 容器代理和 Docker 守护程序使用 HTTP 代理。如果您的容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络，则这非常有用。用于 Linux 和 Windows 实例的过程有所不同，因此请务必阅读下面适用于您的应用程序的部分。

主题

- Amazon Linux 容器实例配置 (p. 245)
- Windows 容器实例配置 (p. 248)

Amazon Linux 容器实例配置

要将 Amazon ECS Linux 容器实例配置为使用 HTTP 代理，请在启动时在相关文件中设置以下变量（利用 Amazon EC2 用户数据）。您也可以手动编辑配置文件并重新启动代理。

`/etc/ecs/ecs.config` (Amazon Linux 2 和 Amazon Linux AMI)

`HTTP_PROXY=10.0.0.131:3128`

将此值设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 ECS 代理用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

`NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

将此值设置为 `169.254.169.254,169.254.170.2,/var/run/docker.sock` 可筛选 EC2 实例元数据、任务的 IAM 角色以及来自代理的 Docker 守护程序流量。

`/etc/systemd/system/ecs.service.d/http-proxy.conf`（仅 Amazon Linux 2）

`Environment="HTTP_PROXY=10.0.0.131:3128/"`

将此值设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 `ecs-init` 代理用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

`Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock"`

将此值设置为 `169.254.169.254,169.254.170.2,/var/run/docker.sock` 可筛选 EC2 实例元数据、任务的 IAM 角色以及来自代理的 Docker 守护程序流量。

`/etc/init/ecs.override`（仅 Amazon Linux AMI）

`env HTTP_PROXY=10.0.0.131:3128`

将此值设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 `ecs-init` 代理用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

`env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

将此值设置为 `169.254.169.254,169.254.170.2,/var/run/docker.sock` 可筛选 EC2 实例元数据、任务的 IAM 角色以及来自代理的 Docker 守护程序流量。

`/etc/systemd/system/docker.service.d/http-proxy.conf`（仅 Amazon Linux 2）

`Environment="HTTP_PROXY=http://10.0.0.131:3128"`

将此值设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 Docker 守护程序用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

`Environment="NO_PROXY=169.254.169.254"`

将此值设置为 `169.254.169.254` 可筛选来自代理的 EC2 实例元数据。

`/etc/sysconfig/docker`（仅限 Amazon Linux AMI 和 Amazon Linux 2）

`export HTTP_PROXY=10.0.0.131:3128`

将此值设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 Docker 守护程序用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

`export NO_PROXY=169.254.169.254,169.254.170.2`

将此值设置为 `169.254.169.254` 可筛选来自代理的 EC2 实例元数据。

在以上文件中设置这些环境变量仅影响 Amazon ECS 容器代理、`ecs-init` 和 Docker 守护程序。它们不配置任何其他服务（如 yum）使用代理。

Example Amazon LinuxHTTP 代理用户数据脚本

下面的示例用户数据 `cloud-boothook` 脚本将 Amazon ECS 容器代理、`ecs-init`、Docker 守护程序和 yum 配置为使用您指定的 HTTP 代理。您还可以指定容器实例在其中自行进行注册的集群。

要在启动容器实例时使用此脚本，请执行启动 Amazon ECS 容器实例 (p. 165)和Step 6.g (p. 166)中的步骤。然后，将下面的 `cloud-boothook` 脚本复制并粘贴到 User data 字段中 (请务必将红色示例值替换为您自己的代理和集群信息)。

> **Note**
>
> 以下用户数据脚本仅支持经过 Amazon ECS 优化的 AMI 的 Amazon Linux 2 和 Amazon Linux AMI 变体。

```
#cloud-boothook
# Configure Yum, the Docker daemon, and the ECS agent to use an HTTP proxy

# Specify proxy host, port number, and ECS cluster name to use
PROXY_HOST=10.0.0.131
PROXY_PORT=3128
CLUSTER_NAME=proxy-test

if grep -q 'Amazon Linux release 2' /etc/system-release ; then
    OS=AL2
    echo "Setting OS to Amazon Linux 2"
elif grep -q 'Amazon Linux AMI' /etc/system-release ; then
    OS=ALAMI
    echo "Setting OS to Amazon Linux AMI"
else
    echo "This user data script only supports Amazon Linux 2 and Amazon Linux AMI."
fi

# Set Yum HTTP proxy
if [ ! -f /var/lib/cloud/instance/sem/config_yum_http_proxy ]; then
    echo "proxy=http://$PROXY_HOST:$PROXY_PORT" >> /etc/yum.conf
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_yum_http_proxy
fi

# Set Docker HTTP proxy (different methods for Amazon Linux 2 and Amazon Linux AMI)
# Amazon Linux 2
if [ $OS == "AL2" ] && [ ! -f /var/lib/cloud/instance/sem/config_docker_http_proxy ]; then
    mkdir /etc/systemd/system/docker.service.d
    cat <<EOF > /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=http://$PROXY_HOST:$PROXY_PORT/"
Environment="HTTPS_PROXY=https://$PROXY_HOST:$PROXY_PORT/"
Environment="NO_PROXY=169.254.169.254,169.254.170.2"
EOF
    systemctl daemon-reload
    if [ "$(systemctl is-active docker)" == "active" ]
    then
        systemctl restart docker
    fi
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_docker_http_proxy
fi
# Amazon Linux AMI
if [ $OS == "ALAMI" ] && [ ! -f /var/lib/cloud/instance/sem/config_docker_http_proxy ];
 then
    echo "export HTTP_PROXY=http://$PROXY_HOST:$PROXY_PORT/" >> /etc/sysconfig/docker
    echo "export HTTPS_PROXY=https://$PROXY_HOST:$PROXY_PORT/" >> /etc/sysconfig/docker
    echo "export NO_PROXY=169.254.169.254,169.254.170.2" >> /etc/sysconfig/docker
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/
config_docker_http_proxy
fi

# Set ECS agent HTTP proxy
if [ ! -f /var/lib/cloud/instance/sem/config_ecs-agent_http_proxy ]; then
    cat <<EOF > /etc/ecs/ecs.config
```

```
ECS_CLUSTER=$CLUSTER_NAME
HTTP_PROXY=$PROXY_HOST:$PROXY_PORT
NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock
EOF
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
agent_http_proxy
fi

# Set ecs-init HTTP proxy (different methods for Amazon Linux 2 and Amazon Linux AMI)
# Amazon Linux 2
if [ $OS == "AL2" ] && [ ! -f /var/lib/cloud/instance/sem/config_ecs-init_http_proxy ];
 then
    mkdir /etc/systemd/system/ecs.service.d
    cat <<EOF > /etc/systemd/system/ecs.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=$PROXY_HOST:$PROXY_PORT/"
Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock"
EOF
    systemctl daemon-reload
    if [ "$(systemctl is-active ecs)" == "active" ]; then
        systemctl restart ecs
    fi
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
init_http_proxy
fi
# Amazon Linux AMI
if [ $OS == "ALAMI" ] && [ ! -f /var/lib/cloud/instance/sem/config_ecs-init_http_proxy ];
 then
    cat <<EOF > /etc/init/ecs.override
env HTTP_PROXY=$PROXY_HOST:$PROXY_PORT
env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock
EOF
    echo "$$: $(date +%s.%N | cut -b1-13)" > /var/lib/cloud/instance/sem/config_ecs-
init_http_proxy
fi
```

Windows 容器实例配置

要将 Amazon ECS Windows 容器实例配置为使用 HTTP 代理，请在启动时设置以下变量（利用 Amazon EC2 用户数据）来达到此目的。

```
[Environment]::SetEnvironmentVariable("HTTP_PROXY",
"http://proxy.mydomain:port", "Machine")
```

将 `HTTP_PROXY` 设置为某个 HTTP 代理的主机名 (或 IP 地址) 和端口号，供 ECS 代理用来连接到 Internet。例如，在容器实例无法通过 Amazon VPC Internet 网关、NAT 网关或实例访问外部网络时。

```
[Environment]::SetEnvironmentVariable("NO_PROXY",
"169.254.169.254,169.254.170.2,\\.\pipe\docker_engine", "Machine")
```

将 `NO_PROXY` 设置为 `169.254.169.254,169.254.170.2,\\.\pipe\docker_engine` 可筛选 EC2 实例元数据、任务的 IAM 角色以及来自代理的 Docker 守护程序流量。

Example Windows HTTP 代理用户数据脚本

下面的示例用户数据 PowerShell 脚本会将 Amazon ECS 容器代理和 Docker 守护程序配置为使用您指定的 HTTP 代理。您还可以指定容器实例在其中自行进行注册的集群。

要在启动容器实例时使用此脚本，请执行步骤 2：在集群中启动 Windows 容器实例 (p. 600)中的步骤。在您到达Step 9 (p. 602) 时，将下面的 PowerShell 脚本复制并粘贴到用户数据字段中 (请务必将红色示例值替换为您自己的代理和集群信息)。

Note

启用任务的 IAM 角色需要使用 `-EnableTaskIAMRole` 选项。有关更多信息，请参阅任务的 Windows IAM 角色 (p. 608)。

```
<powershell>
Import-Module ECSTools

$proxy = "http://proxy.mydomain:port"
[Environment]::SetEnvironmentVariable("HTTP_PROXY", $proxy, "Machine")
[Environment]::SetEnvironmentVariable("NO_PROXY", "169.254.169.254,169.254.170.2,\\.\pipe
\docker_engine", "Machine")

Restart-Service Docker
Initialize-ECSAgent -Cluster MyCluster -EnableTaskIAMRole
</powershell>
```

计划 Amazon ECS 任务

Amazon Elastic Container Service (Amazon ECS) 是一个共享状态的乐观并发系统，可为您的任务和容器提供灵活的计划功能。Amazon ECS 计划程序利用由 Amazon ECS API 提供的相同集群状态信息来制定适当的放置决策。

每个使用 Fargate 启动类型的任务都具有自己的隔离边界，不与其他任务共享底层内核、CPU 资源、内存资源或弹性网络接口。

Amazon ECS 提供的服务计划程序（用于长期运行的任务和应用程序）可以手动运行任务（用于批处理作业或单个运行任务），并通过 Amazon ECS 将任务放置在您的集群上。您可以指定任务放置策略和约束，以允许您在选择的配置中运行任务，例如在各个可用区域上展开任务。还可以集成自定义或第三方计划程序。

服务计划程序

服务计划程序非常适用于长时间运行的无状态服务和应用程序。服务计划程序可确保您指定的计划策略得以遵循并在任务失败时（例如，在底层基础设施因某个原因失败的情况下）重新计划任务。

有两种服务计划程序策略可用：

- `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。
- `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。

  Note

  Fargate 任务不支持 `DAEMON` 计划策略。

此外，服务计划程序可以选择性地确保针对 Elastic Load Balancing 负载均衡器注册任务。您可以更新由服务计划程序维护的服务，例如，部署新的任务定义或更改正在运行的所需任务数。默认情况下，服务计划程序可在多个可用区之间分布任务，但您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 服务 (p. 270)。

手动运行任务

`RunTask` 操作非常适用于诸如执行工作然后停止的批处理作业这样的流程。例如，您可以拥有在工作进入队列时调用 `RunTask` 的流程。任务从队列中拉取工作、执行工作，然后退出。可以使用 `RunTask` 允许默认任务放置策略在集群中随机分配任务，最大程度地减小单一实例获得不成比例数量的任务的机会。您还可以使用 `RunTask` 自定义计划程序如何使用任务放置策略和约束放置任务。有关更多信息，请参阅 Amazon Elastic Container Service API Reference 中的正在运行的任务 (p. 251)和 RunTask。

以类似 **`cron`** 的计划运行任务

如果需要在集群中以设定的时间间隔运行任务，例如备份操作或日志扫描，则可以使用 Amazon ECS 控制台创建 CloudWatch Events 规则，以便在指定的时间在集群中运行一个或多个任务。您可以将计划事件规则设置为特定的时间间隔 (每 *N* 分钟、小时或天运行一次)；而对于更复杂的计划，可以使用 `cron` 表达式。有关更多信息，请参阅排定的任务 (cron) (p. 263)。

自定义计划程序

Amazon ECS 允许您根据业务需要自行创建计划程序，或者利用第三方计划程序。Blox 是一个开源项目，将 Amazon ECS 上运行的应用程序容器化时可以进行更好地控制。它允许您构建计划程序并将第三方计划

程序与 Amazon ECS 集成，同时利用 Amazon ECS 完全管理集群并控制集群规模。自定义计划程序使用 StartTask API 操作将任务置于集群中特定的容器实例上。

> **Note**
>
> 自定义计划程序仅与使用 EC2 启动类型的任务兼容。如果您正在为您的任务使用 Fargate 启动类型，那么 StartTask API 将不起作用。

任务放置

`RunTask` 和 `CreateService` 操作允许您指定任务放置约束和任务放置策略，以自定义 Amazon ECS 放置任务的方式。有关更多信息，请参阅 Amazon ECS 任务放置 (p. 255)。

内容

- 正在运行的任务 (p. 251)
- Amazon ECS 任务放置 (p. 255)
- 排定的任务 (cron) (p. 263)
- 任务生命周期 (p. 265)
- 任务停用 (p. 267)
- Fargate 任务回收 (p. 268)
- 使用 AWS CLI 创建计划任务 (p. 268)

正在运行的任务

在某些情况下，手动运行任务非常适合。例如，假设您正在开发一项任务，但还没有准备好使用服务计划程序来部署这项任务。可能您的任务是一次性或定期批处理作业，并不需要持续运行，完成后也不需要重启。

要使指定数量的任务持续运行或要将任务置于负载均衡器之后，请使用 Amazon ECS 服务计划程序。有关更多信息，请参阅服务 (p. 270)。

内容

- 运行使用 Fargate 启动类型的任务 (p. 251)
- 运行使用 EC2 启动类型的任务 (p. 253)

运行使用 Fargate 启动类型的任务

要运行使用 Fargate 启动类型的任务，请执行以下操作：

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Task Definitions，并选择要运行的任务定义。
   - 要运行此处显示的任务定义的最新修订，请选择要运行的任务定义左侧的框。
   - 要运行此处显示的任务定义的早期修订，请选择任务定义以查看所有有效的修订，然后选择要运行的修订。
3. 选择 Actions，Run Task。
4. 在 Run Task (运行任务) 部分中，完成以下步骤：
   a. 对于 Launch type (启动类型)，选择 FARGATE。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。
   b. 对于平台版本，选择最新。有关平台版本的更多信息，请参阅 AWS Fargate 平台版本 (p. 33)。
   c. 对于 Cluster，选择要使用的集群。
   d. 对于 Number of tasks，键入要使用此任务定义启动的任务的数量。

e. 对于Task Group，键入任务组的名称。

5. 在 VPC and security groups (VPC 和安全组) 部分中，完成以下步骤：

   a. 对于集群 VPC，选择您的任务要使用的 VPC。确保所选的 VPC 未配置为需要专用硬件租赁，因为 Fargate 任务不支持这种方式。

   b. 对于 Subnets，请为您的任务选择可用子网。

   c. 对于 Security groups，已经为任务创建了一个安全组，允许来自 Internet 的 HTTP 流量 (0.0.0.0/0)。要编辑此安全组的名称或规则，或选择现有安全组，请选择 Edit，然后修改安全组设置。

   d. 对于自动分配公有 IP，如果您希望附加到 Fargate 任务的弹性网络接口分配得到一个公有 IP 地址，则选择启用。如果您的任务需要出站网络访问 (例如提取映像)，则这是必需的。如果不需要出站网络访问，则可以选择禁用。

6. 在 Advanced Options (高级选项) 部分中，完成以下步骤：

   - （可选）要将命令、环境变量、任务 IAM 角色或任务执行角色覆盖发送到任务定义中的一个或多个容器，请选择 Advanced Options (高级选项) 并完成以下步骤：

     Note

     如果您将使用任务定义中的参数值，则无需指定覆盖。这些字段仅用于覆盖任务定义中指定的值。

     i. 对于任务角色覆盖，请选择此任务的 IAM 角色以覆盖任务定义中指定的任务 IAM 角色。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

     此处只显示具有 `ecs-tasks.amazonaws.com` 信任关系的角色。有关如何为任务创建 IAM 角色的更多信息，请参阅为任务创建 IAM 角色和策略 (p. 420)。

     ii. 对于任务执行角色覆盖，请选择任务执行角色以覆盖任务定义中指定的任务执行角色。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

     iii. 对于 Container Overrides，选择要接收命令或环境变量覆盖的容器。

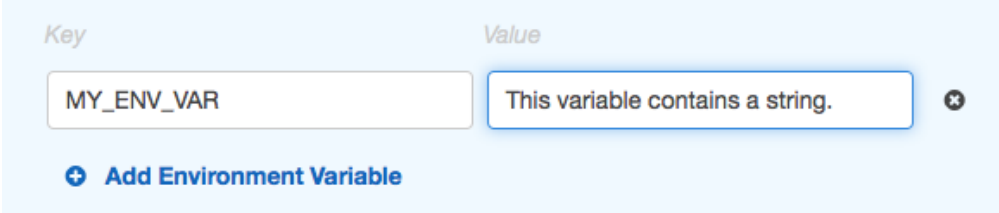
     - 对于命令覆盖：对于 Command override，键入要发送的命令覆盖。如果您的容器定义未指定 `ENTRYPOINT`，则格式应为不带引号的字符串的逗号分隔的列表。例如：

       ```
       /bin/sh,-c,echo,$DATE
       ```

       如果容器定义指定了 `ENTRYPOINT`（如 sh,-c），则格式应为不带引号的字符串，可用双引号将其引起来并作为参数传递给 `ENTRYPOINT` 命令。例如：

       ```
       while true; do echo $DATE > /var/www/html/index.html; sleep 1; done
       ```

     - 对于环境变量覆盖：选择 Add Environment Variable。对于 Key，键入您的环境变量的名称。对于 Value，键入环境值的字符串值 (不使用引号引起来)。

       此环境变量覆盖以如下方式发送到容器：

       ```
       MY_ENV_VAR="This variable contains a string."
       ```

7. In the Task tagging configuration section, complete the following steps:
   a. Select Enable ECS managed tags if you want Amazon ECS to automatically tag each task with the Amazon ECS managed tags. For more information, see Tagging Your Amazon ECS Resources.
   b. For Propagate tags from, select one of the following:
      - Do not propagate – This option will not propagate any tags.
      - Task Definitions – This option will propagate the tags specified in the task definition to the task.

        Note

        If you specify a tag with the same `key` in the Tags section, it will override the tag propagated from the task definition.
8. 在 Tags (标签) 部分中，为要与任务关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
9. 查看您的任务信息并选择 Run Task。

   Note

   如果您的任务从 `PENDING` 移至 `STOPPED`，或者它显示 `PENDING` 状态，然后从列出的任务中消失，则您的任务可能因出错而停止。有关更多信息，请参阅故障排除部分中的检查已停止的任务是否存在错误 (p. 579)。

运行使用 EC2 启动类型的任务

要运行使用 EC2 启动类型的任务，请执行以下操作：

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Task Definitions，并选择要运行的任务定义。
   - 要运行此处显示的任务定义的最新修订，请选择要运行的任务定义左侧的框。
   - 要运行此处显示的任务定义的早期修订，请选择任务定义以查看所有有效的修订，然后选择要运行的修订。
3. 选择 Actions，Run Task。
4. 对于启动类型，选择 EC2。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。
5. 对于 Cluster，选择要使用的集群。对于 Number of tasks，键入要使用此任务定义启动的任务的数量。对于Task Group，键入任务组的名称。
6. 如果任务定义使用 `awsvpc` 网络模式，请完成这些子步骤。否则，请继续下一步。
   a. 对于 Cluster VPC，选择容器实例驻留的 VPC。
   b. 对于 Subnets，请为您的任务选择可用子网。

      Important

      `awsvpc` 网络模式仅支持私有子网。由于任务不接收公有 IP 地址，因此出站 Internet 访问需要 NAT 网关，并且入站 Internet 流量应通过负载均衡器进行路由。
   c. 对于 Security groups，已经为任务创建了一个安全组，允许来自 Internet 的 HTTP 流量 (0.0.0.0/0)。要编辑此安全组的名称或规则，或选择现有安全组，请选择 Edit，然后修改安全组设置。
7. (可选) 对于 Task Placement，您可以使用任务放置策略和约束指定如何放置任务。从以下选项中进行选择：
   - AZ Balanced Spread (AZ 均衡分散) - 在各个可用区以及每个可用区中的各个容器实例中分配任务。
   - AZ Balanced BinPack (AZ 均衡装填) - 在各个可用区以及具有最低可用内存的容器实例中分配任务。
   - BinPack (装填) - 根据 CPU 或内存的最低可用量来分配任务。

- One Task Per Host (每个主机一项任务) - 在每个容器实例中最多可放置服务的一个任务。
- Custom (自定义) - 定义您自己的任务放置策略。有关示例，请参阅 Amazon ECS 任务放置 (p. 255)。

有关更多信息，请参阅 Amazon ECS 任务放置 (p. 255)。

8. （可选）要将命令、环境变量、任务 IAM 角色或任务执行角色覆盖发送到任务定义中的一个或多个容器，请选择高级选项 并完成以下步骤：

   Note

   如果您将使用任务定义中的参数值，则无需指定覆盖。这些字段仅用于覆盖任务定义中指定的值。

   a. 对于任务角色覆盖，请选择此任务的 IAM 角色以覆盖任务定义中指定的任务 IAM 角色。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。

      此处只显示具有 `ecs-tasks.amazonaws.com` 信任关系的角色。有关如何为任务创建 IAM 角色的更多信息，请参阅为任务创建 IAM 角色和策略 (p. 420)。

   b. 对于任务执行角色覆盖，请选择任务执行角色以覆盖任务定义中指定的任务执行角色。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

   c. 对于 Container Overrides，选择要接收命令或环境变量覆盖的容器。

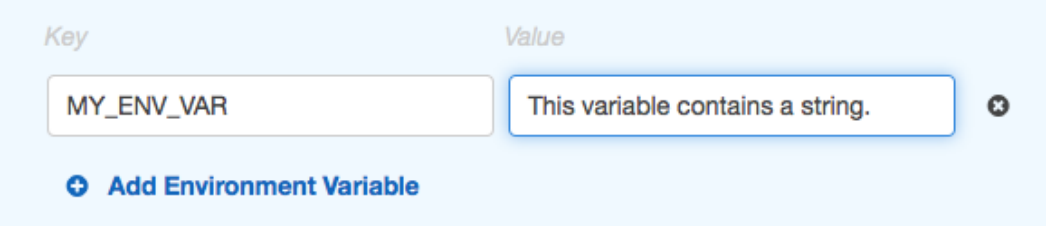
      - 对于命令覆盖：对于 Command override，键入要发送的命令覆盖。如果您的容器定义未指定 `ENTRYPOINT`，则格式应为不带引号的字符串的逗号分隔的列表。例如：

        ```
        /bin/sh,-c,echo,$DATE
        ```

        如果容器定义指定了 `ENTRYPOINT`（如 sh,-c），则格式应为不带引号的字符串，可用双引号将其引起来并作为参数传递给 `ENTRYPOINT` 命令。例如：

        ```
        while true; do echo $DATE > /var/www/html/index.html; sleep 1; done
        ```

      - 对于环境变量覆盖：选择 Add Environment Variable。对于 Key，键入您的环境变量的名称。对于 Value，键入环境值的字符串值 (不使用引号引起来)。

        此环境变量覆盖以如下方式发送到容器：

        ```
        MY_ENV_VAR="This variable contains a string."
        ```

9. In the Task tagging configuration section, complete the following steps:

   a. Select Enable ECS managed tags if you want Amazon ECS to automatically tag each task with the Amazon ECS managed tags. For more information, see Tagging Your Amazon ECS Resources.

   b. For Propagate tags from, select one of the following:

      - Do not propagate – This option will not propagate any tags.
      - Task Definitions – This option will propagate the tags specified in the task definition to the task.

Note

If you specify a tag with the same `key` in the Tags section, it will override the tag propagated from the task definition.

10. 在 Tags (标签) 部分中，为要与任务关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
11. 查看您的任务信息并选择 Run Task。

    Note

    如果您的任务从 `PENDING` 移至 `STOPPED`，或者它显示 `PENDING` 状态，然后从列出的任务中消失，则您的任务可能因出错而停止。有关更多信息，请参阅故障排除部分中的检查已停止的任务是否存在错误 (p. 579)。

Amazon ECS 任务放置

使用 EC2 启动类型来启动任务时，Amazon ECS 必须根据任务定义中指定的要求（例如 CPU 和内存）确定将任务放置在何处。同样，如果您缩减任务计数，Amazon ECS 必须确定终止哪些任务。您可以应用任务放置策略和约束，自定义 Amazon ECS 如何放置和终止任务。使用 Fargate 启动类型的任务不支持任务放置策略和约束。默认情况下，Fargate 任务分布在可用区中。

任务放置策略是一种算法，用于选择放置任务的实例，或要终止的任务。例如，Amazon ECS 可以随机选择实例；还可以选择一组实例，将任务平均分配到其中。

任务放置约束 是放置任务时会考虑的一条规则。例如，您可以使用约束，根据可用区或实例类型来放置任务。您也可以将属性（若干对名称/值）与容器实例相关联，然后根据属性使用约束来放置任务。

Note

任务放置策略是尽力而为。Amazon ECS 仍会尝试放置任务，即使在大多数最优放置选项不可用时也是如此。但是，任务放置约束是绑定的，它们可能阻止任务放置。

您可以将任务放置策略与约束配合使用。例如，您可以在多个可用区中分配任务，并根据每个可用区中的内存装填任务，但只针对 G2 实例。

Amazon ECS 放置任务时，使用以下流程选择容器实例：

1. 标识满足任务定义中对 CPU、内存和端口要求的实例。
2. 识别满足任务放置约束的实例。
3. 识别满足任务放置策略的实例。
4. 选择放置任务的实例。

内容

- Amazon ECS 任务放置策略 (p. 255)
- Amazon ECS 任务放置约束 (p. 257)
- 集群查询语言 (p. 260)

Amazon ECS 任务放置策略

任务放置策略是一种算法，用于选择放置任务的实例，或要终止的任务。可以在运行任务或创建新服务时指定任务放置策略。有关更多信息，请参阅Amazon ECS 任务放置 (p. 255)。

策略类型

Amazon ECS 支持以下任务放置策略：

`binpack`

根据 CPU 或内存的最低可用量来放置任务。这样可尽量减少使用中的实例数量。

`random`

随机放置任务。

`spread`

根据指定值平均放置任务。接受的值为 `instanceId`（或具有相同效果的 `host`）或应用于容器实例的任何平台或自定义属性，如 `attribute:ecs.availability-zone`。服务任务根据该服务的任务分布。独立任务根据同一任务组中的任务分布。

策略示例

您可以使用以下操作指定任务放置策略：CreateService 和 RunTask。

以下策略可在各可用区之间平均分配任务。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    }
]
```

以下策略可在所有实例中平均分配任务。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

以下策略根据内存装填任务。

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

以下策略随机放置任务。

```
"placementStrategy": [
    {
        "type": "random"
    }
]
```

以下策略跨可在多个可用区中平均分配任务，然后在每个可用区内的实例中平均分配任务。

```
"placementStrategy": [
```

```
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread”
    },
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

以下策略可在多个可用区中平均分配任务，然后根据每个可用区内的内存装填任务。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread”
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

Amazon ECS 任务放置约束

任务放置约束 是放置任务时会考虑的一条规则。有关更多信息，请参阅Amazon ECS 任务放置 (p. 255)。

约束类型

Amazon ECS 支持以下类型的任务放置约束：

`distinctInstance`

将每项任务放置在不同的容器实例中。可以在运行任务或创建新服务时指定此任务放置约束。

`memberOf`

将任务放置在满足表达式的容器实例中。有关约束表达式语法的更多信息，请参阅集群查询语言 (p. 260)。

可以使用以下操作指定 `memberOf` 任务放置约束：

- 运行任务
- 创建新服务
- 创建新的任务定义
- 创建现有任务定义的新修订版本

属性

您可以将自定义元数据添加到容器实例中，称为属性。每个属性都有一个名称和一个可选字符串值。您可以使用 Amazon ECS 提供的内置属性，或自定义属性。

内置属性

Amazon ECS 会自动将以下属性应用于您的容器实例。

`ecs.ami-id`

用于启动实例的 AMI 的 ID。本属性的示例值为 "ami-eca289fb"。

`ecs.availability-zone`

实例所在的可用区。本属性的示例值为 "us-east-1a"。

`ecs.instance-type`

实例的实例类型。本属性的示例值为 "g2.2xlarge"。

`ecs.os-type`

实例的操作系统。本属性可能的值为 "linux" 和 "windows"。

自定义属性

您可以将自定义属性应用于您的容器实例。例如，您可以定义名为 "stack"、值为 "prod" 的属性。

添加属性

您可以使用 AWS 管理控制台，在实例注册时使用容器代理添加或手动添加自定义属性。有关使用容器代理的更多信息，请参阅 Amazon ECS 容器代理配置参数 (p. 224)。

使用控制台添加自定义属性

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Clusters 并选择一个集群。
3. 在 ECS Instances 选项卡上，选择容器实例的复选框。
4. 选择 Actions，View/Edit Attributes。
5. 对于每个属性，请执行以下操作：
   a. 选择 Add attribute。
   b. 键入属性的名称和值，然后选中复选标记图标。
6. 属性添加完毕后，选择 Close。

使用 AWS CLI 添加自定义属性

以下示例说明了如何使用 put-attributes 命令添加自定义属性。

示例：单个属性

以下示例将自定义属性 "stack=prod" 添加到默认集群的指定容器实例中。

```
aws ecs put-attributes --attributes name=stack,value=prod,targetId=arn
```

示例：多个属性

以下示例将自定义属性 "stack=prod" 和 "project=a" 添加到默认集群的指定容器实例中。

```
aws ecs put-attributes --attributes name=stack,value=prod,targetId=arn
 name=project,value=a,targetId=arn
```

按属性筛选

您可以为容器实例应用筛选条件，从而能够查看自定义属性。

使用控制台按属性筛选容器实例

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择具有容器实例的集群。

3. 选择 ECS Instances。
4. 通过选择齿轮图标 (⚙) 并选择要显示的属性来设置列可见性首选项。此设置将在与您的账户关联的所有容器集群中保留。
5. 使用 Filter by attributes 文本字段，键入或选择要作为筛选条件的属性。格式必须为 AttributeName:AttributeValue。

   对于 Filter by attributes，键入或选择要作为筛选条件的属性。在选择属性名称后，系统会提示您提供属性值。
6. 根据需要向筛选条件添加其他属性。通过选择某个属性旁的 X 可删除该属性。

使用 AWS CLI 按属性筛选容器实例

以下示例说明了如何使用 list-constainer-instances 命令按属性筛选容器实例。有关筛选语法的更多信息，请参阅集群查询语言 (p. 260)。

示例：内置属性

以下示例使用内置属性列出 g2.2xlarge 实例。

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type == g2.2xlarge"
```

示例：自定义属性

以下示例列出了具有自定义属性 "stack=prod" 的实例。

```
aws ecs list-container-instances --filter "attribute:stack == prod"
```

示例：排除属性值

以下示例列出了具有自定义属性 "stack"，且其值仅为 "prod" 的实例。

```
aws ecs list-container-instances --filter "attribute:stack != prod"
```

示例：多个属性值

以下示例使用内置属性列出 `t2.small` 或 `t2.medium` 类型的实例。

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type in [t2.small,
 t2.medium]"
```

示例：多个属性

以下示例使用内置属性列出 us-east-1a 可用区中的 T2 实例。

```
aws ecs list-container-instances --filter "attribute:ecs.instance-type =~ t2.* and
 attribute:ecs.availability-zone == us-east-1a"
```

任务组

您可以将一组相关任务组成任务组。在执行分布放置时，具有相同任务组名称的所有任务均被视为一组。例如，假设一个集群中运行着不同应用程序，如数据库和 Web 服务器。为确保您的数据库在各可用区之间达到平衡，请将它们添加到名为 "databases" 的任务组，然后使用此任务组约束任务放置。

当您使用 `RunTask` 或 `StartTask` 操作启动任务时，可以为该任务指定任务组名称。如果没有为该任务指定任务组，默认名称是任务定义的系列名称（例如，`family:my-task-definition`）。

对于服务计划程序启动的任务，任务组名称是服务的名称（例如，`service:my-service-name`）。

限制

- 任务组名称必须为 255 个字符或更少。
- 每项任务只能处于一个组中。
- 任务启动后，您将无法修改其任务组。

约束示例

下面是一些任务放置约束示例。

此示例使用 `memberOf` 约束在 T2 实例上放置任务。可以使用以下操作指定此约束：CreateService、RegisterTaskDefinition 和 RunTask。

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

此示例使用 `memberOf` 约束将任务放置在 `databases` 任务组中的实例上。可以使用以下操作指定此约束：CreateService、RegisterTaskDefinition 和 RunTask。

```
"placementConstraints": [
    {
        "expression": "task:group == databases",
        "type": "memberOf"
    }
]
```

`distinctInstance` 约束将组中的每项任务放置于不同实例上。可以使用以下操作指定此约束：CreateService 和 RunTask。

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

集群查询语言

集群查询是允许您将对象分组的表达式。例如，您可以按属性 (例如可用区、实例类型或自定义元数据) 将容器实例分组。有关更多信息，请参阅 属性 (p. 257)。

在您定义了一组容器实例后，可以自定义 Amazon ECS，根据组在容器实例上放置任务。有关更多信息，请参阅 正在运行的任务 (p. 251) 和 创建服务 (p. 317)。您还可以在列出容器实例时应用组筛选条件。有关更多信息，请参阅 按属性筛选 (p. 258)。

表达式语法

表达式有如下语法：

```
subject operator [argument]
```

主题

要评估的属性或字段。

`agentConnected`

按 Amazon ECS 容器代理连接状态选择容器实例。您可以使用此过滤器来搜索具有已断开连接的容器代理的实例。

有效运算符：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`agentVersion`

按 Amazon ECS 容器代理版本选择容器实例。您可以使用此过滤器查找运行过期版本的 Amazon ECS 容器代理的实例。

有效运算符：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`attribute:attribute-name`

按属性选择容器实例。有关更多信息，请参阅属性 (p. 257)。

`ec2InstanceId`

按 Amazon EC2 实例 ID 选择容器实例。

有效运算符：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`registeredAt`

按容器实例注册日期选择容器实例。您可以使用此过滤器查找新注册的实例或已注册很久的实例。

有效运算符：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

有效日期格式：
2018-06-18T22:28:28+00:00、2018-06-18T22:28:28Z、2018-06-18T22:28:28、2018-06-18

`runningTasksCount`

按运行任务数选择容器实例。您可以使用此过滤器查找空或接近空的实例（在其上运行的任务很少）。

有效运算符：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`task:group`

按任务组选择容器实例。有关更多信息，请参阅任务组 (p. 259)。

运算符

比较运算符。支持以下运算符。

运算符	描述
==, equals	字符串相等
!=, not_equals	字符串不相等
>, greater_than	大于
>=, greater_than_equal	大于或等于
<, less_than	小于
<=, less_than_equal	小于或等于
exists	主题存在

运算符	描述
!exists, not_exists	主题不存在
在	值在参数列表中
!in, not_in	值不在参数列表中
=~, matches	模式匹配
!~, not_matches	模式不匹配

> **Note**
>
> 单个表达式不能包含圆括号。但是，可以使用圆括号来指定复合表达式中的优先顺序。

参数

很多运算符的参数是一个文本值。

`in` 和 `not_in` 运算符要求参数是一个参数列表。按如下所示指定参数列表：

```
[argument1, argument2, ..., argumentN]
```

matches 和 not_matches 运算符要求参数符合 Java 正则表达式的语法。有关更多信息，请参阅 java.util.regex.Pattern。

复合表达式

您可以使用以下布尔值运算符组合表达式：

- &&, 和
- ||, 或者
- !, 非

您可以使用圆括号指定优先顺序：

```
(expression1 or expression2) and expression3
```

表达式示例

以下为表达式示例。

示例：字符串相等

以下表达式选择具有指定实例类型的实例。

```
attribute:ecs.instance-type == t2.small
```

示例：参数列表

以下表达式选择在 us-east-1a 或 us-east-1b 可用区中的实例。

```
attribute:ecs.availability-zone in [us-east-1a, us-east-1b]
```

示例：复合表达式

以下表达式选择不在 us-east-1d 可用区内的 G2 实例。

```
attribute:ecs.instance-type =~ g2.* and attribute:ecs.availability-zone != us-east-1d
```

示例：任务关联

以下表达式选择在 `service:production` 组中托管任务的实例。

```
task:group == service:production
```

示例：任务反关联

以下表达式选择未在 database 组中托管任务的实例。

```
not(task:group == database)
```

示例：运行任务数

以下表达式选择仅运行一个任务的实例。

```
runningTasksCount == 1
```

示例：Amazon ECS 容器代理版本

以下表达式选择运行版本低于 1.14.5 的容器代理的实例。

```
agentVersion < 1.14.5
```

示例：实例注册时间

以下表达式选择在 2018 年 2 月 13 日前注册的实例。

```
registeredAt < 2018-02-13
```

示例：Amazon EC2 实例 ID

以下表达式选择具有以下 Amazon EC2 实例 ID 的实例。

```
ec2InstanceId in ['i-abcd1234', 'i-wxyx7890']
```

排定的任务 (cron)

Amazon ECS 支持在类似 `cron` 的计划中或对 CloudWatch Events 的响应中安排任务的能力。使用 Fargate 和 EC2 启动类型的 Amazon ECS 任务支持这一点。

如果需要在集群中以设定的时间间隔运行任务，例如备份操作或日志扫描，则可以使用 Amazon ECS 控制台创建 CloudWatch Events 规则，以便在指定的时间在集群中运行一个或多个任务。您可以将计划事件规则设置为特定的时间间隔 (每 *N* 分钟、小时或天运行一次)；而对于更复杂的计划，可以使用 `cron` 表达式。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式。

您现在也可以将 Fargate 任务设置为 CloudWatch Events 中的任务目标，从而允许您启动任务以响应发生的更改。此外，您可以在使用 `awsvpc` 网络模式时通过 CloudWatch Events 控制台和 AWS CLI 来修改网络配置，这将为 CloudWatch Events 所触发的 Fargate 任务提供与 Amazon EC2 实例相同的联网属性。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的教程：当文件上传至 Amazon S3 存储桶时运行 Amazon ECS 任务。

创建计划任务

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择要在其中创建计划任务的集群。
3. 在 Cluster: ***cluster-name*** 页面上，依次选择 Scheduled Tasks、Create。
4. 在 Schedule rule name 中，为计划规则输入唯一的名称。不得超过 64 个字母、数字、句点、连字符和下划线组合。
5. (可选) 在 Schedule rule description 中，输入规则说明。最多允许 512 个字符。
6. 对于 Schedule rule type，选择计划规则使用固定间隔计划还是 `cron` 表达式。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式 。
   - 对于 Run at fixed interval，为计划输入时间间隔和单位。
   - 对于 Cron expression，为任务计划输入 `cron` 表达式。这些表达式有六个必填字段，字段以空格分隔。有关更多信息及 `cron` 表达式的示例，请参阅 Amazon CloudWatch Events 用户指南 中的 Cron 表达式。
7. 为计划规则创建一个目标。
   a. 对于 Target id (目标 ID)，为目标输入唯一的标识符。不得超过 64 个字母、数字、句点、连字符和下划线组合。
   b. 对于 Launch type (启动类型)，为您的服务中的任务选择启动类型。有关更多信息，请参阅Amazon ECS 启动类型 (p. 81)。
   c. 对于 Task definition，选择要为该目标运行的任务定义的系列和修订 (family:revision)。
   d. 对于 Platform version (平台版本)，选择要用于此目标的平台版本。有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。

      Note

      平台版本仅适用于使用 Fargate 启动类型的任务。

   e. 对于 Number of tasks，输入规则执行时要在集群上运行的指定任务定义的实例化数量。
   f. （可选）对于 Task role override (任务角色覆盖)，选择要为目标中的任务使用的 IAM 角色，而不是任务定义默认值。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。此处仅显示具有 Amazon EC2 Container Service Task Role 信任关系的角色。有关如何为任务创建 IAM 角色的更多信息，请参阅 为任务创建 IAM 角色和策略 (p. 420)。必须为对于 CloudWatch IAM 角色的任何任务角色覆盖添加 `iam:PassRole` 权限。有关更多信息，请参阅CloudWatch Events IAM 角色 (p. 416)。
   g. 如果计划的任务的任务定义使用 `awsvpc` 网络模式，则必须为计划的任务配置 VPC、子网和安全组设置。有关更多信息，请参阅`awsvpc` 网络模式的任务联网 (p. 95)。
      i. 对于 Cluster VPC (集群 VPC)，如果选择 EC2 启动类型，请选择容器实例所在的 VPC。如果选择 Fargate 启动类型，请选择 Fargate 任务应使用的 VPC。确保您选择的 VPC 未配置为需要专用硬件租赁，因为 Fargate 任务对此不予支持。
      ii. 对于 Subnets (子网)，请为计划的任务放置选择可用子网。

         Important

         `awsvpc` 网络模式仅支持私有子网。由于任务不接收公有 IP 地址，因此出站 Internet 访问需要 NAT 网关，并且入站 Internet 流量应通过负载均衡器进行路由。

      iii. 对于 Security groups (安全组)，已经为计划的任务创建了一个安全组，允许来自 Internet 的 HTTP 流量 (`0.0.0.0/0`)。要编辑此安全组的名称或规则，或选择现有安全组，请选择 Edit，然后修改安全组设置。
      iv. 对于自动分配公有 IP，选择是否让您的任务收到公有 IP 地址。如果使用 Fargate 任务，则必须向该任务的弹性网络接口分配一个公有 IP 地址，具有到 Internet 的路由，或可以将请求路由到 Internet 的 NAT 网关。这允许任务提取容器映像。
   h. 对于 CloudWatch Events IAM role for this target (此目标的 CloudWatch Events IAM 角色)，选择已经创建的某个现有 CloudWatch Events 服务角色 (`ecsEventsRole`)。或者，选择 Create new role

(创建新角色) 创建所需的 IAM 角色，以允许 CloudWatch Events 调用 Amazon ECS 代表您运行任务。有关更多信息，请参阅CloudWatch Events IAM 角色 (p. 416)。

> **Important**
>
> 如果计划任务需要使用任务执行角色，或者需要使用任务角色覆盖，则必须将任务执行角色或任务角色覆盖的 `iam:PassRole` 权限添加到 CloudWatch IAM 角色。有关更多信息，请参阅 CloudWatch Events IAM 角色 (p. 416)。

i. (可选) 在 Container overrides 部分中，您可以展开各个容器并覆盖对应容器的命令和/或环境变量 (任务定义中的定义)。

8. (可选) 要添加其他目标 (执行该规则时运行的其他任务)，请选择 Add targets，并为每个其他目标重复上一个子步骤。
9. 选择 Create (创建)。

编辑计划任务

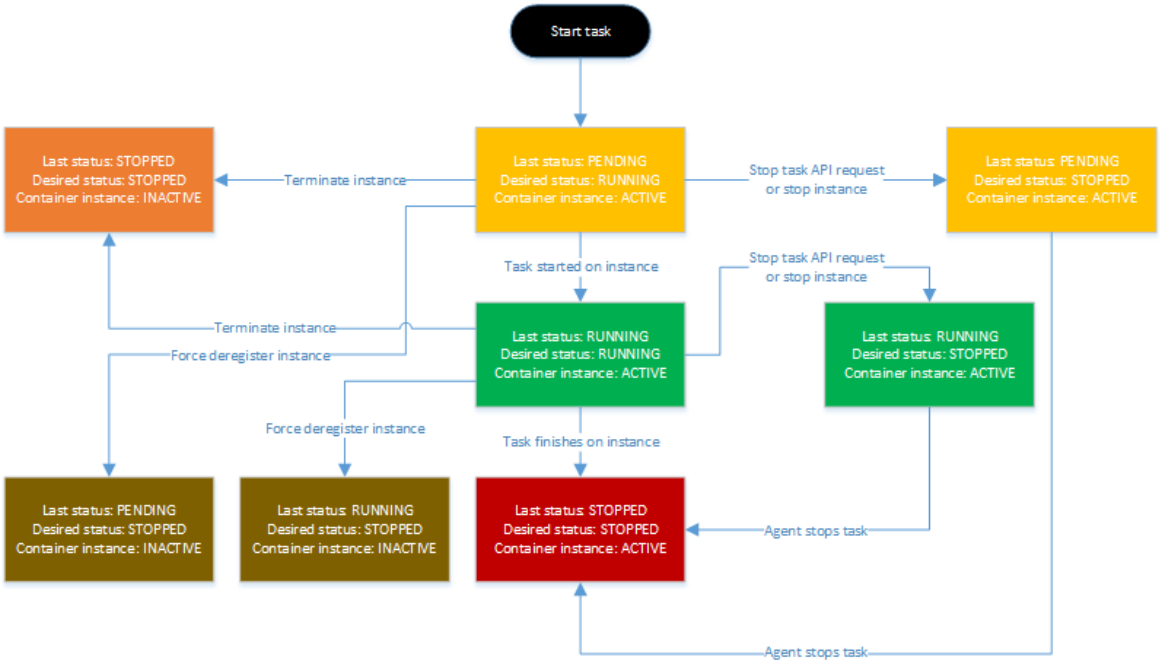
1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择要编辑其计划任务的集群。
3. 在 Cluster: ***cluster-name*** 页面上，选择 Scheduled Tasks。
4. 选中要编辑的计划规则左侧的框，然后选择 Edit。
5. 编辑要更新的字段，然后选择 Update。

任务生命周期

如果手动启动任务或将任务作为服务的一部分启动，则任务在自行完成或被手动停止之前可能经历多种状态。一些任务旨在作为批处理作业运行，将自然地依次经历 `PENDING`、`RUNNING` 和 `STOPPED` 状态。可为服务的一部分的其他任务旨在无限期持续运行，或根据需要向上扩展和向上扩展。

当请求任务状态更改（如停止任务或更新所需服务计数以将其扩展或缩减）时，Amazon ECS 容器代理会根据任务的上一个已知状态 (`lastStatus`) 和任务的所需状态 (`desiredStatus`) 跟踪这些更改。任务的上一个已知状态和所需状态均可在控制台中或通过使用 API 或 AWS CLI 描述任务进行查看。

以下流程图显示任务生命周期流程。



生命周期状态

以下是每个任务生命周期状态的描述。

PROVISIONING (正在预置)

Amazon ECS 必须先执行其他步骤，然后再启动任务。例如，对于使用 `awsvpc` 网络模式的任务，需要预置弹性网络接口。

PENDING (待处理)

这是一个过渡状态，其中 Amazon ECS 正在等待容器代理采取进一步操作。

ACTIVATING (正在激活)

Amazon ECS 必须在启动任务后但在任务可以过渡到 `RUNNING` 状态之前执行其他步骤。例如，对于已配置服务发现的任务，必须创建服务发现资源。对于配置为使用多个 Elastic Load Balancing 目标组的服务所包含的任务，在此状态下将注册目标组。

RUNNING (正在运行)

任务正在成功运行。

DEACTIVATING (正在停用)

Amazon ECS 必须先执行其他步骤，然后再停止任务。例如，对于配置为使用多个 Elastic Load Balancing 目标组的服务所包含的任务，在此状态下将注册目标组。

STOPPING (正在停止)

这是一个过渡状态，其中 Amazon ECS 正在等待容器代理采取进一步操作。

DEPROVISIONING (正在取消预置)

Amazon ECS 必须在已停止任务后但在任务过渡到 `STOPPED` 状态之前执行其他步骤。例如，对于使用 `awsvpc` 网络模式的任务，需要分离并删除弹性网络接口。

STOPPED (已停止)

任务已成功停止。

任务停用

Amazon ECS 任务停用将影响 Fargate 和 EC2 启动类型，并且您将收到待处理停用的通知电子邮件。

可计划在以下场景中停用任务：

- AWS 检测到托管任务的底层硬件存在无法修复的故障。
- 您的任务使用 Fargate 并在具有安全漏洞的平台版本上运行，这需要您使用修补后的平台版本启动新任务来替换任务。

如果已计划停用任务，您将在事件发生之前收到包含任务 ID 和停用日期的电子邮件。该电子邮件将发送至与您的账户关联的地址，也就是您用于登录 AWS 管理控制台的电子邮件地址。如果您使用的是不会定期检查的电子邮件账户，则可以使用 AWS 个人运行状况控制面板确定是否有计划停用的任务。要更新您的账户的联系人信息，请转到 Account Settings (账户设置) 页面。

当任务到达其计划的停用日期时，AWS 会将其停止或终止。如果任务是服务的一部分，则任务会自动停止，服务计划程序会启动新任务来代替它。如果您使用的是独立任务，则会收到任务停用通知，您必须启动新任务来替换它们。

使用计划停用的任务

如果任务是服务的一部分，则任务会自动停止，服务计划程序会在任务到达计划的停用日期后启动新任务来替换它。如果需要在停用日期前更新服务任务，可以按照以下步骤操作。有关更多信息，请参阅 更新服务 (p. 326)。

更新运行服务（AWS 管理控制台）

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏上，选择集群所在的区域。
3. 在导航窗格中，选择 Clusters。
4. 在集群页面上，选择您的服务驻留的集群的名称。
5. 在 Cluster: ***name*** 页面上，选择 Services。
6. 选中要更新的服务左侧的框并选择 Update。
7. 在配置服务页面上，您的服务信息已预先填充。选择 Force new deployment (强制实施新部署)，然后选择 Next step (下一步)。

   Note

   对于使用 Fargate 启动类型的任务，强制实施新部署将使用修补后的平台版本启动新任务。您的任务不需要您选择其他平台版本。有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。
8. 在 Configure network (配置网络) 和 Set Auto Scaling (optional) (设置 Auto Scaling (可选)) 页面上，选择 Next step (下一步)。
9. 选择 Update Service 以完成并更新服务。

更新运行服务（AWS CLI）

1. 获取服务的 ARN。

```
aws ecs list-services --cluster cluster_name --region region
```

   输出:

```
{
```

```
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/MyService"
    ]
}
```

2. 更新服务，强制实施将部署新任务的新部署。

```
aws ecs update-service --service serviceArn --force-new-deployment --
cluster cluster_name --region region
```

如果您使用的是独立任务，则可以启动新的任务来代替它。有关更多信息，请参阅 正在运行的任务 (p. 251)。

Fargate 任务回收

Amazon ECS 任务回收仅影响使用 Fargate 的任务，并且在回收事件前不会发送通知。

在以下场景中，任务可以回收：

- 任务使用 Fargate 启动类型并使用平台版本 1.3.0 或更高版本。有关更多信息，请参阅 AWS Fargate 平台版本 (p. 33)。

  Note

  使用版本 1.3.0 之前的平台的 Fargate 任务不受影响。
- 任务是 Amazon ECS 服务的一部分。独立任务不受任务回收的影响，但可能仍被计划停用。有关更多信息，请参阅 任务停用 (p. 267)。
- AWS 确定存在回收任务的原因，如下所述。

当 AWS 确定 Fargate 任务需要安全或基础设施更新时，它会对任务应用所需的补丁。其中大多数补丁是透明的，任务无需停止，但有时需要回收任务。从 Fargate 平台版本 1.3.0 开始，作为服务的一部分启动的任何 Fargate 任务可能会停止并由 Amazon ECS 服务计划程序启动新任务，来为任务提供尽可能最佳的安全性和可用性。任务回收在 2019 年 2 月 1 日之后开始并将依据滚动机制继续。服务计划程序将确保保持您的服务的所需任务计数。

为了准备好采用这一新过程，建议通过模拟此方案来测试您的应用程序行为。通过停止您的服务中的单个任务来测试复原能力，可以实现此目的。

使用 AWS CLI 创建计划任务

本主题演示如何使用 AWS CLI 创建计划任务。创建计划的任务使用 CloudWatch Events API。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的什么是 Amazon CloudWatch Events？。

完成以下先决任务：

- 设置 AWS 账户。
- 安装和配置 AWS CLI。有关更多信息，请参阅 AWS 命令行界面。

创建计划的任务

1. 创建 CloudWatch Events 规则。该示例创建一个名为 `MyRule1` 的可在协调世界时每天下午的 12:00 触发的规则。

```
aws events put-rule --schedule-expression "cron(0 12 * * ? *)" --name MyRule1
```

> **Note**
>
> 有关规则表达式的其他示例，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式。

2. 添加 ECS 集群的详细信息和任务定义作为 CloudWatch Events 规则的目标。使用完整 ARN 指定集群和任务定义。

   此示例将 `MyRule1` 的目标定义为 `default` 集群中的 `first-run-task-definition:1` 任务定义并为其分配 `ecsEventsRole` IAM 角色。它请求对 `1` 任务进行计划。必须已创建集群和任务定义。否则，您会收到错误。

```
aws events put-targets --rule "MyRule1" --targets "Id"="1","Arn"="arn:aws:ecs:us-
east-1:123456789012:cluster/default","RoleArn"="arn:aws:iam::123456789012:role/
ecsEventsRole","EcsParameters"="{"TaskDefinitionArn"= "arn:aws:ecs:us-
east-1:123456789012:task-definition/first-run-task-definition:1","TaskCount"= 1}"
```

服务

利用 Amazon ECS，您可在 Amazon ECS 集群中同时运行和管理指定数量的任务定义实例。这称为服务。如果您的任何任务应该出于任何原因失败或停止，Amazon ECS 服务计划程序将启动另一个任务定义实例来替换它并根据所用的计划策略在服务中保留预期数量的任务。

除了在服务中保留预期数量的任务之外，您还可选择借助负载均衡器运行您的服务。负载均衡器将在与服务关联的各个任务间分配流量。

主题：

- 服务计划程序概念 (p. 270)
- 附加服务概念 (p. 271)
- 服务定义参数 (p. 272)
- Amazon ECS 部署类型 (p. 277)
- 服务负载均衡 (p. 284)
- 服务 Auto Scaling (p. 299)
- 服务发现 (p. 306)
- 创建服务 (p. 317)
- 更新服务 (p. 326)
- 删除服务 (p. 328)
- 服务限制逻辑 (p. 329)

服务计划程序概念

如果服务中的某个任务停止，表明该任务已终止，一个新的任务已启动。此过程将继续，直到您的服务根据您指定的计划策略达到所需的运行任务数。

服务计划程序包括用来限制任务反复启动失败时重启任务的频率的逻辑。如果一个任务尚未进入 `RUNNING` 状态便停止 (取决于具有 `startedAt` 时间戳的任务)，则服务计划程序开始以增量方式减缓启动尝试，并会发出一条服务事件消息。此行为可防止不必要的资源用于失败的任务上，从而让您有机会解决问题。服务更新后，服务计划程序恢复正常行为。有关更多信息，请参阅 服务限制逻辑 (p. 329) 和 服务事件消息 (p. 581)。

有两种服务计划程序策略可用：

- `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。
- `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。

  Note

  Fargate 任务不支持 `DAEMON` 计划策略。

守护程序

守护程序计划策略只在每个活动容器实例上部署一个任务，以满足集群中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。

守护程序服务计划程序不会将任何任务放置在具有 DRAINING 状态的实例上。如果容器实例转换为 DRAINING，则会停止其上的守护程序任务。它还监视何时将新容器实例添加到您的集群，并将守护程序任务添加到其中。

如果指定了 deploymentConfiguration，则最大百分比参数必须为 100。maximumPercent 的守护程序服务的默认值为 100%。minimumHealthyPercent 的守护程序服务的默认值对于 AWS CLI、AWS 开发工具包和 API 为 0%，对于 AWS 管理控制台为 50%。

> **Note**
>
> 守护程序服务计划程序不支持使用 传统负载均衡器。

副本

副本计划策略在集群上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务，但您可以使用任务放置策略和约束自定义任务放置决策。

当服务计划程序（使用 REPLICA 策略）启动新任务或停止运行使用 Fargate 启动类型的任务时，它会尝试在服务中的可用区之间保持平衡。

当服务计划程序（使用 REPLICA 策略）使用 EC2 启动类型启动新任务时，计划程序使用以下逻辑：

- 确定您的集群中可支持您的服务的任务定义的容器实例（例如，它们有必需的 CPU、内存、端口和容器实例属性）。
- 确定哪些容器实例满足为该服务定义的任何放置约束。
- 如果已定义放置策略，请使用该策略来从剩余候选项中选择一个实例。
- 如果尚未定义放置策略，请使用以下逻辑对集群中各个可用区的任务进行均衡：
  - 对有效容器实例进行排序，从而优先考虑在各自的可用区中具有此服务的运行任务数量最少的实例。例如，如果 A 区有一个正在运行的服务任务，B 区和 C 区都没有正在运行的服务任务，则认为在 B 区或 C 区中的有效容器实例中最适合放置任务。
  - 将新的服务任务放在最佳可用区中的有效容器实例上（基于之前的步骤），并优先使用具有此服务的最小数量的运行任务的容器实例。

当服务计划程序（使用 REPLICA 策略）停止运行任务时，它将尝试在集群的可用区之间保持均衡。对于使用 EC2 启动类型的任务，计划程序使用以下逻辑：

- 如果已定义放置策略，请使用该策略来选择要终止的任务。例如，如果某服务已定义了可用区分布策略，则选择一项任务终止，以让剩下的任务保持最佳分布。
- 如果尚未定义放置策略，请使用以下逻辑来为集群中的各个可用区保持均衡：
  - 对有效容器实例进行排序，从而优先考虑在各自的可用区中具有此服务的运行任务数量最多的实例。例如，如果 A 区有 1 个正在运行的服务任务，B 区和 C 区各有 2 个正在运行的服务任务，则认为 B 区或 C 区中的容器实例最适合终止任务。
  - 在最佳可用区中的容器实例上停止任务（基于之前的步骤），并优先使用具有此服务的最大数量的运行任务的容器实例。

附加服务概念

- 您可选择借助负载均衡器运行您的服务。有关更多信息，请参阅 服务负载均衡 (p. 284)。
- 您可选择为您的服务指定部署配置。在部署（通过更新任务定义或预期数量的服务触发）期间，服务计划程序将使用最小和最大正常运行百分比参数确定部署策略。有关更多信息，请参阅服务定义参数 (p. 272)。
- 您可以选择配置服务以使用 Amazon ECS 服务发现。服务发现 使用 Amazon Route 53 自动命名 API 来管理服务任务的 DNS 条目，使其可在 VPC 中供搜索。有关更多信息，请参阅服务发现 (p. 306)。

- 在删除服务后，如果仍有需要清除的运行任务，服务状态将从 `ACTIVE` 变为 `DRAINING`，并且服务将不再在控制台或 ListServices API 操作中可见。所有任务都变为 `STOPPING` 或 `STOPPED` 状态后，服务状态将从 `DRAINING` 变为 `INACTIVE`。处于 `DRAINING` 或 `INACTIVE` 状态的服务仍可使用 DescribeServices API 操作查看。但在未来，`INACTIVE` 服务可能从 Amazon ECS 记录中被清除，于是这些服务上的 DescribeServices 调用将返回 `ServiceNotFoundException` 错误。

服务定义参数

服务定义将定义要用于您的服务的任务定义、该任务要运行的实例化的数量、要与您的任务关联的负载均衡器（如果有）以及其他服务参数。

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "ENABLED"
        }
```

```
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "ECS"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
}
```

Note

您可使用以下 AWS CLI 命令创建上述服务定义模板。

```
aws ecs create-service --generate-cli-skeleton
```

您可在服务定义中指定以下参数。

`cluster`

要在其上运行您的服务的集群的短名称或完整 Amazon 资源名称 (ARN)。如果您未指定集群，则采用 `default` 集群。

`serviceName`

您的服务的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。 一个集群中的服务不得重名，但您可在一个区域或多个区域的多个集群中拥有名称相似的服务。

必需：是

`taskDefinition`

要在您的服务中运行的任务定义的 `family` 和 `revision` (`family:revision`) 或完整 Amazon 资源名称 (ARN)。如果未指定 `revision`，将使用最新的 `ACTIVE` 修订。

`loadBalancers`

表示要用于您的服务的负载均衡器的负载均衡器对象。对于使用 应用程序负载均衡器 或 网络负载均衡器 的服务，最多可将 5 个目标组附加到一个服务。

.在创建服务后，在服务定义中指定的负载均衡器名称或目标组 ARN、容器名称和容器端口是不可变的。

对于 传统负载均衡器，此对象必须包含负载均衡器名称、容器名称 (与在容器定义中显示的相同) 和容器端口才能从负载均衡器进行访问。将此服务中的任务放在容器实例上后，容器实例将注册到在此处指定的负载均衡器。

对于应用程序负载均衡器和网络负载均衡器，此对象必须包含负载均衡器目标组 ARN、容器名称（与在容器定义中显示的相同）和容器端口才能从负载均衡器进行访问。将此服务中的任务放在容器实例上后，容器实例和端口组合将注册为在此处指定的目标组中的目标。

`targetGroupArn`

与服务关联的 Elastic Load Balancing 目标组的完整 Amazon 资源名称 (ARN)。

`loadBalancerName`

负载均衡器的名称。

`containerName`

要与负载均衡器关联的容器的名称（与在容器定义中显示的相同）。

`containerPort`

要与负载均衡器关联的容器上的端口。此端口必须对应于服务的任务定义中的 `containerPort`。您的容器实例必须允许端口映射的 `hostPort` 上的入口流量。

`serviceRegistries`

服务的服务发现配置的详细信息。有关更多信息，请参阅服务发现 (p. 306)。

`registryArn`

服务注册表的 Amazon 资源名称 (ARN)。当前支持的服务注册表是 Amazon Route 53 自动命名。有关更多信息，请参阅服务。

`port`

在服务发现服务指定了一个 SRV 记录时使用的端口值。如果使用 `awsvpc` 网络模式和 SRV 记录，则需要此字段。

`containerName`

已在任务定义中指定的容器名称值，用于服务发现服务。如果服务任务指定的任务定义使用 `bridge` 或 `host` 网络模式，则必须从任务定义中指定 `containerName` 和 `containerPort` 组合。如果服务任务指定的任务定义使用 `awsvpc` 网络模式，并且使用类型 SRV DNS 记录，则必须指定 `containerName` 和 `containerPort` 组合或 `port` 值，但不能同时指定二者。

`containerPort`

已在任务定义中指定的端口值，用于服务发现服务。如果服务任务指定的任务定义使用 `bridge` 或 `host` 网络模式，则必须从任务定义中指定 `containerName` 和 `containerPort` 组合。如果服务任务指定的任务定义使用 `awsvpc` 网络模式，并且使用类型 SRV DNS 记录，则必须指定 `containerName` 和 `containerPort` 组合或 `port` 值，但不能同时指定二者。

`desiredCount`

要在您的集群中放置并保持运行的指定任务定义的实例化数量。

`clientToken`

您为确保请求的幂等性而提供的唯一、区分大小写的标识符。最多允许 32 个 ASCII 字符。

`launchType`

运行您的服务的启动类型。接受的值为 `FARGATE` 或 `EC2`。如果没有指定启动类型，将默认使用 `EC2`。有关更多信息，请参阅Amazon ECS 启动类型 (p. 81)。

`platformVersion`

正在运行服务中任务的平台版本。仅为使用 Fargate 启动类型的任务指定平台版本。如果没有指定任何版本，将默认使用最新版本 (`LATEST`)。

AWS Fargate 平台版本用于指代 Fargate 任务基础设施的特定运行时环境。在运行任务或创建服务的过程中指定 `LATEST` 平台版本时，您将获得可用于任务的最新平台版本。当您扩展服务时，这些任务将收到在服务的当前部署中指定的平台版本。有关更多信息，请参阅 AWS Fargate 平台版本 (p. 33)。

> **Note**
>
> 不会使用 EC2 启动类型为任务指定平台版本。

`role`

允许 Amazon ECS 代表您调用负载均衡器的 IAM 角色的名称或完整 ARN。仅当您在服务中使用负载均衡器且任务定义未使用 `awsvpc` 网络模式时，才允许使用此参数。如果您指定 `role` 参数，还必须指定具有 `loadBalancers` 参数的负载均衡器对象。

如果您的指定角色的路径并非 `/`，则必须指定完整角色 ARN（推荐）或将此路径作为角色名称的前缀。例如，如果名称为 `bar` 的角色的路径为 `/foo/`，您应指定 `/foo/bar` 作为角色名称。有关更多信息，请参阅 IAM 用户指南 中的易记名称和路径。

> **Important**
>
> 如果您的账户已创建 Amazon ECS 服务相关角色，则默认情况下会为您的服务使用该角色，除非您在此处指定一个角色。如果您的任务定义使用 awsvpc 网络模式（在这种情况下，您不应在此处指定角色），则需要服务相关角色。有关更多信息，请参阅对 Amazon ECS 使用服务相关角色 (p. 392)。

`deploymentConfiguration`

可选部署参数，用于控制部署期间运行的任务数以及停止和开始任务的顺序。

`maximumPercent`

如果服务使用滚动更新 (`ECS`) 部署类型，`maximumPercent` 参数以 `desiredCount` 的百分比形式表示您的服务中在部署期间允许处于 `RUNNING` 或 `PENDING` 状态的任务数的上限（向下取整到最近的整数）。此参数使您能够定义部署批次大小。例如，如果您的服务使用 `REPLICA` 服务计划程序且任务的 `desiredCount` 为 4，`maximumPercent` 值为 200%，则计划程序将在停止 4 个旧任务之前开始 4 个新任务（前提是具有执行此操作所需的集群资源）。使用 `REPLICA` 服务计划程序的服务的默认 `maximumPercent` 值为 200%。

如果您的服务使用的是 `DAEMON` 服务计划程序类型，则 `maximumPercent` 应保持在 100%，这是默认值。

部署期间的最大任务数为 `desiredCount` 乘以 `maximumPercent`/100（向下取整到最近的整数值）。

如果服务使用的是蓝/绿 (`CODE_DEPLOY`) 或 `EXTERNAL` 部署类型和使用 EC2 启动类型的任务，最大百分比值将设置为默认值并在容器实例处于 `DRAINING` 状态时用于定义服务中保持在 `RUNNING` 状态的任务数的上限。如果服务中的任务使用 Fargate 启动类型，将不会使用最大百分比值，尽管该值会在描述您的服务时返回。

`minimumHealthyPercent`

如果服务使用滚动更新 (`ECS`) 部署类型，`minimumHealthyPercent` 以 `desiredCount` 的百分比形式表示您的服务中在部署期间必须保持在 `RUNNING` 状态的任务数的下限（向上取整到最近的整数）。此参数使您不必使用额外的集群容量就能部署。例如，如果您的服务的任务 `desiredCount` 为 4，`minimumHealthyPercent` 为 50%，则该服务计划程序可能在开始两个新任务之前停止两个现有任务以释放集群容量。

服务任务的状态如果为 `RUNNING`，而且未 使用负载均衡器，即为运行正常。

在以下情况下，使用负载均衡器的服务的任务被视为运行正常：处于 `RUNNING` 状态、已通过所有已定义的运行状况检查、在负载均衡器或目标组上报告为正常运行。

`minimumHealthyPercent` 的副本服务的默认值在 AWS 管理控制台中为 50%，对于 AWS CLI、AWS 开发工具包和 API 为 100%。使用 `DAEMON` 服务计划的默认 `minimumHealthyPercent` 值对于 AWS CLI、AWS 开发工具包和 API 为 0%，对于 AWS 管理控制台为 50%。

部署期间的最小正常任务数为 `desiredCount` 乘以 `minimumHealthyPercent`/100（向上取整到最近的整数值）。

如果服务使用的是蓝/绿 (`CODE_DEPLOY`) 或 `EXTERNAL` 部署类型和使用 EC2 启动类型的任务，最小正常百分比值将设置为默认值并在容器实例处于 `DRAINING` 状态时用于定义服务中保持在 `RUNNING` 状态的任务数的下限。如果服务中的任务使用 Fargate 启动类型，将不会使用最小正常运行百分比值，尽管该值会在描述您的服务时返回。

`placementConstraints`

您的服务中的任务使用的一组放置约束对象。对于每个任务，您可以指定多达 10 种约束 (此限制包括任务定义中的约束和这些在运行时指定的约束)。如果您使用的是 Fargate 启动类型，则不支持任务放置约束。

`type`

约束类型。使用 `distinctInstance` 确保特定组中的每个任务在不同的容器实例上运行。使用 `memberOf` 将选择限制为一组有效的候选。任务定义不支持值 `distinctInstance`。

`expression`

应用于约束的集群查询语言表达式。请注意，如果约束类型为 `distinctInstance`，您不能指定表达式。有关更多信息，请参阅 集群查询语言 (p. 260)。

`placementStrategy`

您的服务中的任务所用的放置策略对象。对于每项服务，您最多可以指定 4 种策略。

`type`

放置策略的类型。`random` 放置策略将任务随机放置在可用的候选上。`spread` 放置策略基于 `field` 参数将任务均匀地放置在可用候选上。`binpack` 策略将任务放置在可用资源数量最少（使用 `field` 参数指定）的可用候选上。例如，如果对内存使用装填，则任务会放置到剩余内存最少但仍够运行该任务的实例上。

`field`

应用放置策略的字段。对于 `spread` 放置策略，有效值为 `instanceId` (或具有相同效果的 `host`) 或应用于容器实例的任何平台或自定义属性，如 `attribute:ecs.availability-zone`。对于 `binpack` 放置策略，有效值为 `cpu` 和 `memory`。对于 `random` 放置策略，该字段未用。

`networkConfiguration`

服务的网络配置。对于使用 `awsvpc` 网络模式接收其自己的弹性网络接口的任务定义，此参数是必需的，其他网络模式不支持此参数。如果使用 Fargate 启动类型，则需要 `awsvpc` 网络模式。有关更多信息，请参阅`awsvpc` 网络模式的任务联网 (p. 95)。

`awsvpcConfiguration`

表示任务或服务的子网和安全组的对象。

`subnets`

与任务或服务相关联的子网。

`securityGroups`

与任务或服务相关联的安全组。如果您未指定安全组，将使用 VPC 的默认安全组。

`assignPublicIP`

任务的弹性网络接口是否接收公有 IP 地址。

`healthCheckGracePeriodSeconds`

在任务首次启动后，Amazon ECS 服务计划程序应忽略不正常的 Elastic Load Balancing 目标运行状况检查、容器运行状况检查和 Route 53 运行状况检查的时间段（以秒为单位）。您的服务只有配置为使用负载均衡器时此设置才适用。如果您的服务的任务需要一段时间才能启动和响应运行状况检查，则您最长可以指定 2,147,483,647 秒的运行状况检查宽限期，在此期间，ECS 服务计划程序将忽略运行状况检查状态。此宽限期可防止 ECS 服务计划程序将由于时间不足尚未启动的任务标记为不正常并停止它们。

`schedulingStrategy`

要使用的计划策略。有关更多信息，请参阅 服务计划程序概念 (p. 270)。

有两种服务计划程序策略可用：

- `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。
- `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。

  Note

  Fargate 任务不支持 `DAEMON` 计划策略。

`deploymentController`

要用于该服务的部署控制器。有关更多信息，请参阅 Amazon ECS 部署类型 (p. 277)。

`type`

要使用的部署控制器类型。部署控制器有三种类型：

`ECS`

滚动更新 (`ECS`) 部署类型涉及将当前运行版本的容器替换为最新版本。Amazon ECS 在滚动更新期间在服务中添加或删除的容器数量可以通过调整服务部署期间允许的运行正常的任务的最小和最大数量进行控制，如 deploymentConfiguration 中指定。

`CODE_DEPLOY`

蓝/绿 (`CODE_DEPLOY`) 部署类型使用 CodeDeploy 提供技术支持的蓝/绿部署模型，从而让您可以先验证服务的新部署，然后再向其发送生产流量。

`EXTERNAL`

外部部署类型让您可以使用任何第三方部署控制器以便完全控制 Amazon ECS 服务的部署过程。

`tags`

您应用于服务以帮助您对其进行分类和组织的元数据。每个标签都包含您定义的一个键和一个可选值。在服务被删除时，标签也会随之被删除。标签键的最大字符长度为 128 个字符，标签值的最大长度为 256 个字符。有关更多信息，请参阅 标记您的 Amazon ECS 资源 (p. 330)。

`key`

构成标签的键-值对的一个部分。键是一种常见的标签，行为类似于更具体的标签值的类别。

`value`

构成标签的键-值对的可选部分。值充当标签类别（键）中的描述符。

`enableECSManagedTags`

指定是否要为服务中的任务启用 Amazon ECS 托管标签。有关更多信息，请参阅标记资源以便于计费 (p. 332)。

`propagateTags`

指定是否要将标签从任务定义或服务复制到服务中的任务。如果未指定值，则不会复制标签。只能在创建服务的过程中将标签复制到服务中的任务。要在创建服务以后将标签添加到任务，请使用 `TagResource` API 操作。

Amazon ECS 部署类型

Amazon ECS 部署类型决定了您的服务使用的部署策略。有三种部署类型：滚动更新、蓝/绿和外部。

主题

- 滚动更新 (p. 278)
- 使用 CodeDeploy 进行蓝/绿部署 (p. 278)
- 外部部署 (p. 281)

滚动更新

滚动更新 部署类型由 Amazon ECS 控制。这涉及将当前运行版本的容器替换为最新版本的服务计划程序。在滚动更新期间 Amazon ECS 在服务中添加或删除的任务数量由部署配置控制。部署配置包括在服务部署期间允许的任务的最小数量和最大数量。

要创建使用滚动更新部署类型的新 Amazon ECS 服务，请参阅创建服务 (p. 317)。

使用 CodeDeploy 进行蓝/绿部署

蓝/绿 部署类型使用由 CodeDeploy 控制的蓝/绿部署模型。此部署类型让您可以先验证服务的新部署，然后再向其发送生产流量。有关更多信息，请参阅 AWS CodeDeploy User Guide 中的什么是 CodeDeploy？。

以下是在服务使用蓝/绿部署类型时，Amazon ECS 使用的 CodeDeploy 的组件：

CodeDeploy 应用程序

CodeDeploy 资源的集合。这包括一个或多个部署组。

CodeDeploy 部署组

部署设置。这包括以下内容：

- Amazon ECS 集群和服务
- 负载均衡器目标组和侦听器信息
- 部署回滚策略
- 流量重新路由设置
- 原始修订终止设置
- 部署配置
- 可设置为停止部署的 CloudWatch 警报配置
- SNS 或 CloudWatch Events 通知设置

有关更多信息，请参阅 AWS CodeDeploy User Guide 中的使用部署组。

CodeDeploy 部署配置

指定在部署期间 CodeDeploy 如何将生产流量路由到您的替换任务集。目前唯一支持的值为 `CodeDeployDefault.AllAtOnce`，这意味着所有流量同时从原始任务集路由到替换任务集。有关更多信息，请参阅 AWS CodeDeploy User Guide 中的使用部署配置。

修订

修订是 CodeDeploy 应用程序规范文件（AppSpec 文件）。在 AppSpec 文件中，指定任务定义的完整 ARN 以及在创建新部署时要将流量路由到的替换任务集的容器和端口。容器名称必须是任务定义中引用的容器名称之一。如果在服务定义中更新了网络配置或平台版本，则还必须在 AppSpec 文件中指定这些详细信息。您也可以指定要在部署生命周期事件期间运行的 Lambda 函数。Lambda 函数允许您在部署期间运行测试和返回指标。有关更多信息，请参阅 AWS CodeDeploy User Guide 中的 AppSpec 文件参考。

蓝/绿部署注意事项

使用蓝/绿部署类型时，请考虑以下内容：

- 在初次创建使用蓝/绿部署类型的 Amazon ECS 服务时，将创建一个 Amazon ECS 任务集。
- 您必须将该服务配置为使用 应用程序负载均衡器 或 网络负载均衡器。不支持 传统负载均衡器。以下是负载均衡器要求：
  - 您必须将生产侦听器添加到用于路由生产流量的负载均衡器。
  - 可以将可选的测试侦听器添加到负载均衡器，这用于路由测试流量。如果您指定了测试侦听器，则在部署期间 CodeDeploy 会将测试流量路由到替换任务集。
  - 生产和测试侦听器必须属于同一负载均衡器。
  - 您必须为负载均衡器定义一个目标组。目标组通过生产侦听器将流量路由到服务中的原始任务集。
- 使用蓝/绿部署类型时，不支持服务 Auto Scaling。
- 当您最初创建 CodeDeploy 应用程序和部署组时，必须指定以下内容：
  - 您必须为负载均衡器定义两个目标组。一个目标组应为在创建 Amazon ECS 服务时为负载均衡器定义的初始目标组。第二个目标组的唯一要求是，它不能与服务所使用的负载均衡器之外的其他负载均衡器相关联。
- 在为 Amazon ECS 服务创建 CodeDeploy 部署后，CodeDeploy 会在该部署中创建替换任务集（或绿色任务集）。如果您将测试侦听器添加到了负载均衡器，则 CodeDeploy 会将测试流量路由到替换任务集。此时您可以运行任何验证测试。然后，CodeDeploy 会根据部署组的流量重新路由设置将生产流量从原始任务集重新路由到替换任务集。

Amazon ECS 控制台体验

Amazon ECS 控制台中的服务创建和服务更新工作流程支持蓝/绿部署。

要创建使用蓝/绿部署类型的 Amazon ECS 服务，请参阅创建服务 (p. 317)。

要更新使用蓝/绿部署类型的现有 Amazon ECS 服务，请参阅更新服务 (p. 326)。

在使用 Amazon ECS 控制台创建使用蓝/绿部署类型的 Amazon ECS 服务时，将自动使用以下默认设置创建 Amazon ECS 任务集和以下 CodeDeploy 资源。

资源	默认设置
应用程序名称	`AppECS-<cluster[:47]>-<service[:47]>`
部署组名称	`DgpECS-<cluster[:47]>-<service[:47]>`
部署组负载均衡器信息	将负载均衡器生产侦听器、可选的测试侦听器和指定的目标组添加到部署组配置。
流量重新路由设置	流量重新路由 – 默认设置为 Reroute traffic immediately (立即重新路由流量)。您可以在 CodeDeploy 控制台上或通过更新 `TrafficRoutingConfig` 来更改该设置。有关更多信息，请参阅 AWS CodeDeploy API Reference 中的 CreateDeploymentConfig。
原始修订终止设置	原始修订终止设置配置为在重新路由流量之后等待 1 小时再终止蓝色任务集。
部署配置	部署配置设置为 `CodeDeployDefault.AllAtOnce`，这会一次性地将所有流量从蓝色任务集路由到绿色任务集。您无法更改此设置。
自动回滚配置	如果部署失败，自动回滚设置配置为将其回滚。

要查看使用蓝/绿部署类型的 Amazon ECS 服务的详细信息，请使用 Amazon ECS 控制台上的 Deployments (部署) 选项卡。

要在 CodeDeploy 控制台中查看 CodeDeploy 部署组的详细信息，请参阅 AWS CodeDeploy User Guide 中的使用 CodeDeploy 查看部署组详细信息。

要在 CodeDeploy 控制台中修改 CodeDeploy 部署组的设置，请参阅 AWS CodeDeploy User Guide 中的使用 CodeDeploy 更改部署组设置。

蓝/绿部署所需的 IAM 权限

通过组合使用 Amazon ECS 和 CodeDeploy API 可进行 Amazon ECS 蓝/绿部署。IAM 用户必须对这些服务拥有适当权限，然后才能在 AWS 管理控制台中或通过 AWS CLI 或开发工具包使用 Amazon ECS 蓝/绿部署。

除了用于创建和更新服务的标准 IAM 权限之外，Amazon ECS 还需要以下权限。这些权限已添加到 `AmazonECS_FullAccess` IAM 策略。有关更多信息，请参阅AmazonECS_FullAccess (p. 397)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateApplication",
        "codedeploy:CreateDeployment",
        "codedeploy:CreateDeploymentGroup",
        "codedeploy:GetApplication",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentGroup",
        "codedeploy:ListApplications",
        "codedeploy:ListDeploymentGroups",
        "codedeploy:ListDeployments",
        "codedeploy:StopDeployment",
        "codedeploy:GetDeploymentTarget",
        "codedeploy:ListDeploymentTargets",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision",
        "codedeploy:BatchGetApplicationRevisions",
        "codedeploy:BatchGetDeploymentGroups",
        "codedeploy:BatchGetDeployments",
        "codedeploy:BatchGetApplications",
        "codedeploy:ListApplicationRevisions",
        "codedeploy:ListDeploymentConfigs",
        "codedeploy:ContinueDeployment",
        "sns:ListTopics",
        "cloudwatch:DescribeAlarms",
        "lambda:ListFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Note

除了运行任务和服务所需的标准 Amazon ECS 权限外，IAM 用户还需要 `iam:PassRole` 权限，以使用 IAM 角色执行任务。

CodeDeploy 需要调用 Amazon ECS API、修改 Elastic Load Balancing、调用 Lambda 函数和描述 CloudWatch 警报的权限，以及代表您修改服务的预期数量的权限。在创建使用蓝/绿部署类型的 Amazon ECS 服务之前，您必须先创建一个 IAM 角色 (`ecsCodeDeployRole`)。有关更多信息，请参阅 Amazon ECS CodeDeploy IAM 角色 (p. 411)。

创建服务示例 (p. 383)和更新服务示例 (p. 384) IAM 策略示例说明 IAM 用户在 AWS 管理控制台上使用 Amazon ECS 蓝/绿部署所需的权限。

外部部署

外部 部署类型使您能够使用任何第三方部署控制器来完全控制 Amazon ECS 服务的部署过程。服务的详细信息由服务管理 API 操作（`CreateService`、`UpdateService` 和 `DeleteService`）或任务集管理 API 操作（`CreateTaskSet`、`UpdateTaskSet`、`UpdateServicePrimaryTaskSet` 和 `DeleteTask`）管理。每个 API 操作都有其可以管理一部分服务定义参数。

`UpdateService` API 操作更新服务的预期数量和运行状况检查宽限期参数。如果需要更新启动类型、平台版本、负载均衡器详细信息、网络配置或任务定义，您必须创建一个新任务集。

`UpdateTaskSet` API 操作仅更新任务集的扩展参数。

`UpdateServicePrimaryTaskSet` API 操作修改服务中的哪个任务集是主要任务集。当您调用 `DescribeServices` API 操作时，它将返回为主要任务集指定的所有字段。如果更新服务的主要任务集，新的主要任务集上现有的任何任务集参数值若不同于服务中旧的主要任务集的参数，则在定义新的主要任务集时，这些参数将会更新为新值。如果没有为服务定义任何主要任务集，则在描述服务时，任务集字段将为 null。

> **Important**
>
> 使用外部部署控制器时，不支持服务 Auto Scaling。

以下是在 Amazon ECS 上管理外部部署的基本工作流程。

使用外部部署控制器管理 Amazon ECS 服务

1. 创建 Amazon ECS 服务。唯一必需的参数是服务名称。您可以在使用外部部署控制器创建服务时指定以下参数。在服务内创建任务集时将指定所有其他服务参数。

   `serviceName`

   您的服务的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。 一个集群中的服务不得重名，但您可在一个区域或多个区域的多个集群中拥有名称相似的服务。

   必需：是

   `desiredCount`

   要在该服务内放置并保持运行的指定任务集任务定义的实例化数量。

   `deploymentConfiguration`

   可选部署参数，用于控制部署期间运行的任务数以及停止和开始任务的顺序。有关更多信息，请参阅 deploymentConfiguration。

   `tags`

   您应用于服务以帮助您对其进行分类和组织的元数据。每个标签都包含您定义的一个键和一个可选值。在服务被删除时，标签也会随之被删除。标签键的最大字符长度为 128 个字符，标签值的最大长度为 256 个字符。有关更多信息，请参阅 标记您的 Amazon ECS 资源 (p. 330)。

   `key`

   构成标签的键-值对的一个部分。键是一种常见的标签，行为类似于更具体的标签值的类别。

`value`

构成标签的键-值对的可选部分。值充当标签类别（键）中的描述符。

`enableECSManagedTags`

指定是否要为该服务内的任务启用 Amazon ECS 托管标签。有关更多信息，请参阅标记资源以便于计费 (p. 332)。

`propagateTags`

指定是否要将标签从任务定义或服务复制到服务中的任务。如果未指定值，则不会复制标签。只能在创建服务的过程中将标签复制到服务中的任务。要在创建服务以后将标签添加到任务，请使用 `TagResource` API 操作。

`healthCheckGracePeriodSeconds`

在任务首次启动后，Amazon ECS 服务计划程序应忽略不正常的 Elastic Load Balancing 目标运行状况检查、容器运行状况检查和 Route 53 运行状况检查的时间段（以秒为单位）。您的服务只有配置为使用负载均衡器时此设置才适用。如果您的服务的任务需要一段时间才能启动和响应运行状况检查，则您最长可以指定 2,147,483,647 秒的运行状况检查宽限期，在此期间，ECS 服务计划程序将忽略运行状况检查状态。此宽限期可防止 ECS 服务计划程序将由于时间不足尚未启动的任务标记为不正常并停止它们。

`schedulingStrategy`

要使用的计划策略。使用外部部署控制器的服务仅支持 `REPLICA` 计划策略。有关更多信息，请参阅服务计划程序概念 (p. 270)。

`placementConstraints`

您的服务中的任务使用的一组放置约束对象。对于每个任务，您可以指定多达 10 种约束（此限制包括任务定义中的约束和这些在运行时指定的约束）。如果您使用的是 Fargate 启动类型，则不支持任务放置约束。

`placementStrategy`

您的服务中的任务所用的放置策略对象。对于每项服务，您最多可以指定 4 种策略。

下面是使用外部部署控制器的服务的示例服务定义。

```
{
    "cluster": "",
    "serviceName": "",
    "desiredCount": 0,
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
```

```
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
}
```

2. 创建初始任务集。任务集包含有关您的服务的以下详细信息：

   `taskDefinition`

   要使用的任务集中的任务的任务定义。

   `launchType`

   运行您的服务的启动类型。接受的值为 `FARGATE` 或 `EC2`。如果没有指定启动类型，将默认使用 `EC2`。有关更多信息，请参阅Amazon ECS 启动类型 (p. 81)。

   `platformVersion`

   正在运行服务中任务的平台版本。仅为使用 Fargate 启动类型的任务指定平台版本。如果没有指定任何版本，将默认使用最新版本 (`LATEST`)。

   AWS Fargate 平台版本用于指代 Fargate 任务基础设施的特定运行时环境。在运行任务或创建服务的过程中指定 `LATEST` 平台版本时，您将获得可用于任务的最新平台版本。当您扩展服务时，这些任务将收到在服务的当前部署中指定的平台版本。有关更多信息，请参阅 AWS Fargate 平台版本 (p. 33)。

   > **Note**
   >
   > 不会使用 EC2 启动类型为任务指定平台版本。

   `loadBalancers`

   表示要用于您的服务的负载均衡器的负载均衡器对象。当使用外部部署控制器时，仅支持 应用程序负载均衡器 和 网络负载均衡器。如果您使用的是 应用程序负载均衡器，每个任务集仅允许一个 应用程序负载均衡器 目标组。

   `networkConfiguration`

   服务的网络配置。对于使用 `awsvpc` 网络模式接收其自己的弹性网络接口的任务定义，该参数是必需的，其他网络模式不支持该参数。有关更多信息，请参阅`awsvpc` 网络模式的任务联网 (p. 95)。

   `serviceRegistries`

   要分配给此服务的服务发现注册表的详细信息。有关更多信息，请参阅服务发现 (p. 306)。

   `scale`

   要在任务集中放置并保持运行的任务的预期数量的浮点百分比。该值以服务的 `desiredCount` 的总数的百分比的形式指定。接受的值为 0 到 100 之间的数字。

3. 需要服务更改时，请使用 `UpdateService`、`UpdateTaskSet` 或 `CreateTaskSet` API 操作，具体取决于您要更新的参数。如果创建了任务集，请使用服务中每个任务集的 `scale` 参数以确定要在服务中保持运行的任务数量。例如，如果您有一个包含 `tasksetA` 的服务并且创建一个 `tasksetB`，则您可能在希望向其过渡生产流量之前测试 `tasksetB` 的有效性。您可以将两个任务集的 `scale` 设置为 `100`，并且在您已准备好将所有生产流量过渡到 `tasksetB` 时，您可以将 `tasksetA` 的 `scale` 更新为 `0` 以缩小规模。

服务负载均衡

您可以选择将 Amazon ECS 服务配置为使用 Elastic Load Balancing 跨服务中的任务平均分配流量。

Amazon ECS 服务支持 应用程序负载均衡器、网络负载均衡器 和 传统负载均衡器 负载均衡器类型。应用程序负载均衡器 用于路由 HTTP/HTTPS（或第 7 层）流量。网络负载均衡器 和 传统负载均衡器 用于路由 TCP（或第 4 层）流量。有关更多信息，请参阅 负载均衡器类型 (p. 285)。

应用程序负载均衡器提供了一些新功能，这使其非常适合用于 Amazon ECS 服务：

- 每项服务可通过指定多个目标组来为来自多个负载均衡器的流量提供服务并公开多个负载均衡端口。
- 它们受使用 Fargate 和 EC2 启动类型的任务的支持。
- 应用程序负载均衡器 允许容器使用动态主机端口映射（以便每个容器实例允许来自同一服务的多个任务）。
- 应用程序负载均衡器支持基于路径的路由和优先级规则（以便多个服务可使用一个应用程序负载均衡器上的相同侦听器端口）。

我们建议对 Amazon ECS 服务使用应用程序负载均衡器，以便利用这些最新功能，除非服务需要只有网络负载均衡器或 传统负载均衡器才提供的功能。有关 Elastic Load Balancing 以及负载均衡器类型之间的差异的更多信息，请参阅 Elastic Load Balancing 用户指南。

主题

- 负载均衡概念 (p. 284)
- 负载均衡器类型 (p. 285)
- 检查您的账户的服务角色 (p. 288)
- 创建负载均衡器 (p. 289)
- 向服务注册多个目标组 (p. 298)

负载均衡概念

- 对于使用 应用程序负载均衡器 或 网络负载均衡器 的服务，最多可将 5 个目标组附加到一个服务。
- 具有使用 `awsvpc` 网络模式的任务（例如，那些 Fargate 启动类型的任务）的服务仅支持 应用程序负载均衡器 和 网络负载均衡器；不支持 传统负载均衡器。此外，当您为这些服务创建任何目标组时，必须选择 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。
- 如果使用 应用程序负载均衡器 的服务需要访问多个负载均衡端口（例如，端口 80 和端口 443 用于 HTTP/HTTPS 服务），则可以配置两个侦听器：一个负责 HTTPS 的侦听器，它将请求转发给服务；另一个负责将 HTTP 请求重定向到相应 HTTPS 端口的侦听器。有关更多信息，请参阅 Application Load Balancer 用户指南 中的创建您的 应用程序负载均衡器 的侦听器。
- 在一个任务定义中启动的所有容器始终放置在相同的容器实例上。对于 传统负载均衡器，您可以选择通过在服务定义中定义多个主机端口并将这些侦听器端口添加到负载均衡器，来将多个容器（位于相同的任务定义中）放在同一负载均衡器后面。例如，如果任务定义包含在容器实例上使用端口 3030 的 Elasticsearch 以及在容器实例上使用端口 4040 的 Logstash 和 Kibana，则同一负载均衡器可通过两个侦听器将流量路由到 Elasticsearch 和 Kibana。有关更多信息，请参阅 Classic Load Balancer 用户指南 中的 传统负载均衡器的侦听器 。

  Important

  建议不要将多项服务连接到同一个 传统负载均衡器。因为整个容器实例是通过 传统负载均衡器（而不是主机和端口组合）注册和取消注册的，所以，此配置会在一个服务中的任务停止时导致出现问题，从而导致整个容器实例从 传统负载均衡器取消注册，而相同容器实例上的来自不同

服务的另一个任务仍将使用它。如果您要将多个服务连接到一个负载均衡器（例如，为了节省成本），建议您使用 应用程序负载均衡器。

- 如果任务属于配置为使用 传统负载均衡器 的服务的一部分，则此类任务不支持容器运行状况检查。
- 您的负载均衡器子网配置必须包含容器实例所在的所有可用区。
- 在创建服务后，在服务定义中指定的目标组 ARN 或负载均衡器名称、容器名称和容器端口是不可变的。您无法添加、删除或更改现有服务的负载均衡器配置。如果您更新服务的任务定义，则创建服务时指定的容器名称和容器端口必须保留在任务定义中。
- 如果服务的任务未达到负载均衡器运行状况检查标准，则系统会终止并重启该任务。此过程将持续到您的服务达到预期的运行任务的数量。
- 如果您的支持负载均衡器的服务出现问题，请参阅服务负载均衡器问题排查 (p. 587)。

负载均衡器类型

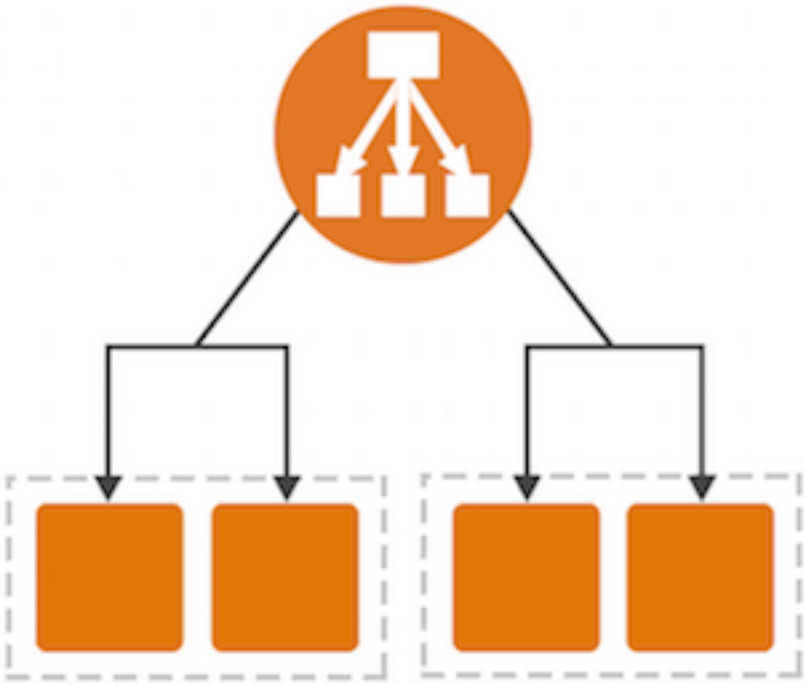
Elastic Load Balancing 支持以下类型的负载均衡器：应用程序负载均衡器、网络负载均衡器 和 传统负载均衡器。 Amazon ECS 服务可使用任一类型的负载均衡器。 应用程序负载均衡器 用于路由 HTTP/HTTPS（即第 7 层）流量。网络负载均衡器 和 传统负载均衡器 用于 TCP（即第 4 层）流量。

主题

- 应用程序负载均衡器 (p. 285)
- 网络负载均衡器 (p. 286)
- 传统负载均衡器 (p. 287)

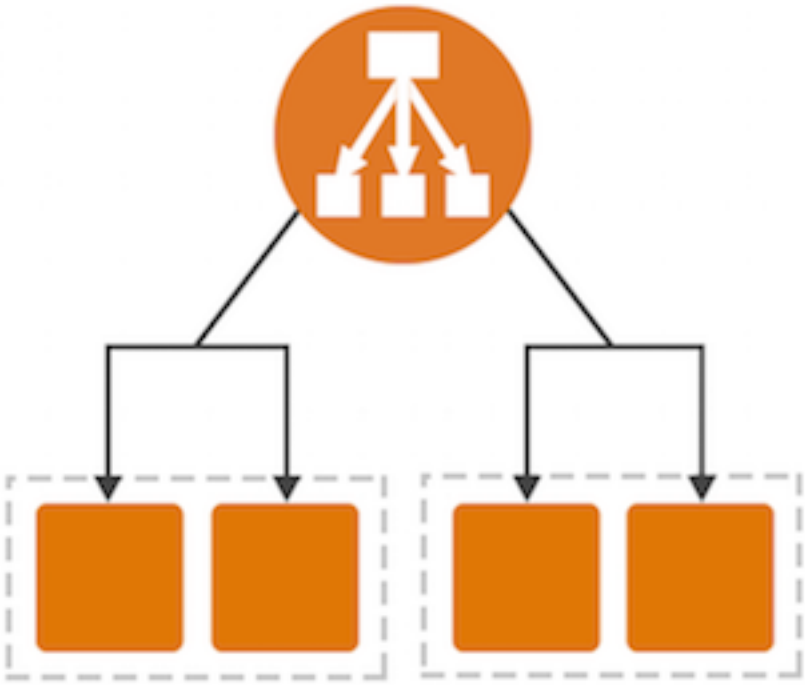
应用程序负载均衡器

应用程序负载均衡器 在应用程序层 (HTTP/HTTPS) 作出路由决策，支持基于路径的路由，并且可以将请求路由到集群中每个容器实例上的一个或多个端口。应用程序负载均衡器 支持动态主机端口映射。例如，如果任务的容器定义指定端口 80 为 NGINX 容器端口，并指定端口 0 为主机端口，则从容器实例的临时端口范围（例如，在最新的Amazon ECS-optimized AMI 上，为 32768 到 61000）中动态选择主机端口。在启动任务时，NGINX 容器将作为实例 ID 和端口组合注册到 应用程序负载均衡器，并且流量将分配到与该容器对应的实例 ID 和端口。此动态映射可让您在同一容器实例上拥有来自单个服务的多个任务。有关更多信息，请参见 Application Load Balancer 用户指南。



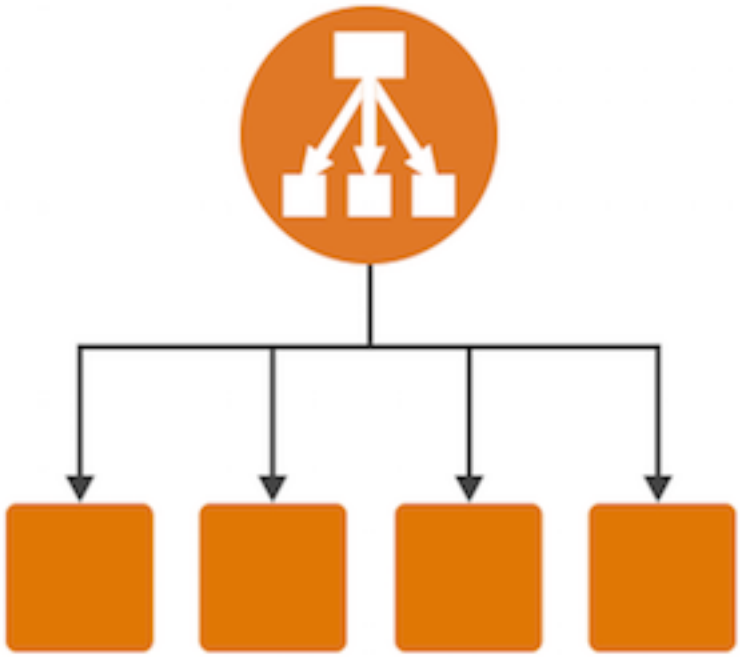
网络负载均衡器

网络负载均衡器 在传输层 (TCP/SSL) 制定路由决策。它每秒可以处理数百万个请求。在负载均衡器收到连接后，它会使用流式哈希路由算法从目标组中选择一个目标作为默认规则。它尝试在侦听器配置中指定的端口上打开一个到该选定目标的 TCP 连接。它转发请求，而不修改标头。网络负载均衡器 支持动态主机端口映射。例如，如果任务的容器定义指定端口 80 为 NGINX 容器端口，并指定端口 0 为主机端口，则从容器实例的临时端口范围（例如，在最新的Amazon ECS-optimized AMI 上，为 32768 到 61000）中动态选择主机端口。在启动任务时，NGINX 容器将作为实例 ID 和端口组合注册到 网络负载均衡器，并且流量将分配到与该容器对应的实例 ID 和端口。此动态映射可让您在同一容器实例上拥有来自单个服务的多个任务。有关更多信息，请参见 Network Load Balancer 用户指南。



传统负载均衡器

传统负载均衡器 在传输层 (TCP/SSL) 或应用程序层 (HTTP/HTTPS) 制定路由决策。传统负载均衡器 当前需要负载均衡器端口与容器实例端口间的固定关系。例如，可以将负载均衡器端口 80 映射到容器实例端口 3030，将负载均衡器端口 4040 映射到容器实例端口 4040。但是，无法将负载均衡器端口 80 映射到一个容器实例上的端口 3030 和另一个容器实例上的端口 4040。此静态映射需要您的集群至少具有与使用 传统负载均衡器 的单一服务的预期数量一样多的容器实例。有关更多信息，请参见 Classic Load Balancer 用户指南。



检查您的账户的服务角色

在创建和停止任务时，Amazon ECS 需要向负载均衡器注册和取消注册容器实例的权限。

在大多数情况下，Amazon ECS 服务角色是在 Amazon ECS 控制台首次运行体验中自动创建的。您可使用以下过程检查并确定您的账户是否已有 Amazon ECS 服务角色。

在 IAM 控制台中检查 **ecsServiceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsServiceRole`。如果该角色不存在，请参阅 Amazon ECS 服务计划程序 IAM 角色 (p. 410)以创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions。
5. 在 Managed Policies 部分，确保将 AmazonEC2ContainerServiceRole 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS 服务角色。否则，请执行以下子步骤来附加策略。
   a. 选择 Attach Policy。
   b. 对于 Filter，键入 **`AmazonEC2ContainerServiceRole`** 以缩小要附加的可用策略的范围。
   c. 选中 AmazonEC2ContainerServiceRole 策略左侧的框并选择 Attach Policy。
6. 选择信任关系，然后选择编辑信任关系。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2008-10-17",
  "Statement": [
```

```
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建负载均衡器

本节提供有关通过 AWS 管理控制台将 Elastic Load Balancing 用于 Amazon ECS 服务的实践经验介绍。在本节中，您将创建一个外部负载均衡器，该外部负载均衡器接收公共网络流量并将其路由到 Amazon ECS 容器实例。

Elastic Load Balancing 支持以下类型的负载均衡器：应用程序负载均衡器、网络负载均衡器和 传统负载均衡器。Amazon ECS 服务可使用任一类型的负载均衡器。应用程序负载均衡器用于路由 HTTP/HTTPS 流量。网络负载均衡器和 传统负载均衡器用于路由 TCP（即第 4 层）流量。

应用程序负载均衡器提供了一些新功能，这使其非常适合用于 Amazon ECS 服务：

- 应用程序负载均衡器 允许容器使用动态主机端口映射（以便每个容器实例允许来自同一服务的多个任务）。
- 应用程序负载均衡器支持基于路径的路由和优先级规则（以便多个服务可使用一个应用程序负载均衡器上的相同侦听器端口）。

建议将应用程序负载均衡器用于 Amazon ECS 服务，以便利用这些最新功能。有关 Elastic Load Balancing 以及负载均衡器类型之间的差异的更多信息，请参阅 Elastic Load Balancing 用户指南。

主题

- 创建 应用程序负载均衡器 (p. 289)
- 创建 网络负载均衡器 (p. 293)
- 创建 传统负载均衡器 (p. 294)

创建 应用程序负载均衡器

本节将指导您完成在 AWS 管理控制台中创建应用程序负载均衡器的过程。

定义您的负载均衡器

首先，为负载均衡器提供一些基本配置信息（如名称、网络和侦听器）。

侦听器是用于检查连接请求的进程。使用前端 (客户端到负载均衡器) 连接的协议和端口与后端 (负载均衡器到后端实例) 连接的协议和端口配置侦听器。在此示例中，您将配置一个侦听器，该侦听器接受端口 80 上的 HTTP 请求并使用 HTTP 将它们发送至端口 80 上的任务中的容器。

定义您的负载均衡器

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中，选择负载均衡器的区域。请确保选择您为 Amazon ECS 容器实例选择的同一个区域。
3. 在导航窗格中的 LOAD BALANCING 下，选择 Load Balancers。
4. 选择 Create Load Balancer。

5. 在 Select load balancer type (选择负载均衡器类型) 页面上，选择 应用程序负载均衡器，然后选择 Continue (继续)。
6. 完成 Configure Load Balancer 页面，如下所示：
   a. 对于 Name，键入负载均衡器的名称。
   b. 对于 Scheme (模式)，面向 Internet 的负载均衡器将来自客户端的请求通过 Internet 路由到目标。内部负载均衡器使用私有 IP 地址将请求路由到目标。
   c. 对于 IP address type，选择 ipv4 可仅支持 IPv4 地址，选择 dualstack 可同时支持 IPv4 和 IPv6 地址。
   d. 对于 Listeners，默认值是负责接收端口 80 上的 HTTP 流量的侦听器。您可以保留默认侦听器设置，修改侦听器的协议或端口，或者选择 Add 以添加另一个侦听器。

      Note

      如果您计划将流量路由至多个目标组，请参阅 ListenerRules 以获取有关如何添加基于主机或路径的规则的详细信息。
   e. 对于 VPC，选择您用于要在其上运行服务的容器实例的相同 VPC。
   f. 对于 Availability Zones，选中要为负载均衡器启用的可用区的复选框。如果该可用区有一个子网，则将选择此子网。如果该可用区有多个子网，请选择其中一个子网。您只能为每个可用区域选择一个子网。您的负载均衡器子网配置必须包含容器实例所在的所有可用区。
   g. 选择 Next: Configure Security Settings。

（可选）配置安全设置

如果您在上一步骤中创建了安全侦听器，请按完成 Configure Security Settings 页面（如下所示）；否则，请选择 Next: Configure Security Groups。

配置安全设置

1. 如果有来自 AWS Certificate Manager 的证书，则选择 Choose an existing certificate from AWS Certificate Manager (ACM) (从 AWS Certificate Manager (ACM)选择现有证书)，然后从 Certificate name (证书名称) 选择该证书。
2. 如果已使用 IAM 上传了证书，则选择 Choose an existing certificate from AWS Identity and Access Management (IAM) (从 AWS Identity and Access Management (IAM)选择现有证书)，然后从 Certificate name (证书名称) 选择证书。
3. 如果您有准备好的证书可以上传，则选择 Upload a new SSL Certificate to AWS Identity and Access Management (IAM)。对于 Certificate name，键入证书的名称。对于 Private Key，复制并粘贴私钥文件 (PEM 编码的文件) 的内容。在 Public Key Certificate 中，复制并粘贴公钥证书文件（PEM 编码的文件）的内容。在 Certificate Chain 中，复制并粘贴证书链文件（PEM 编码的文件）的内容，除非您使用的是自签名证书并且浏览器是否隐式接受证书并不重要。
4. 对于 Select policy，选择一个预定义的安全策略。有关安全策略的详细信息，请参阅安全策略。
5. 选择 Next: Configure Security Groups。

配置安全组

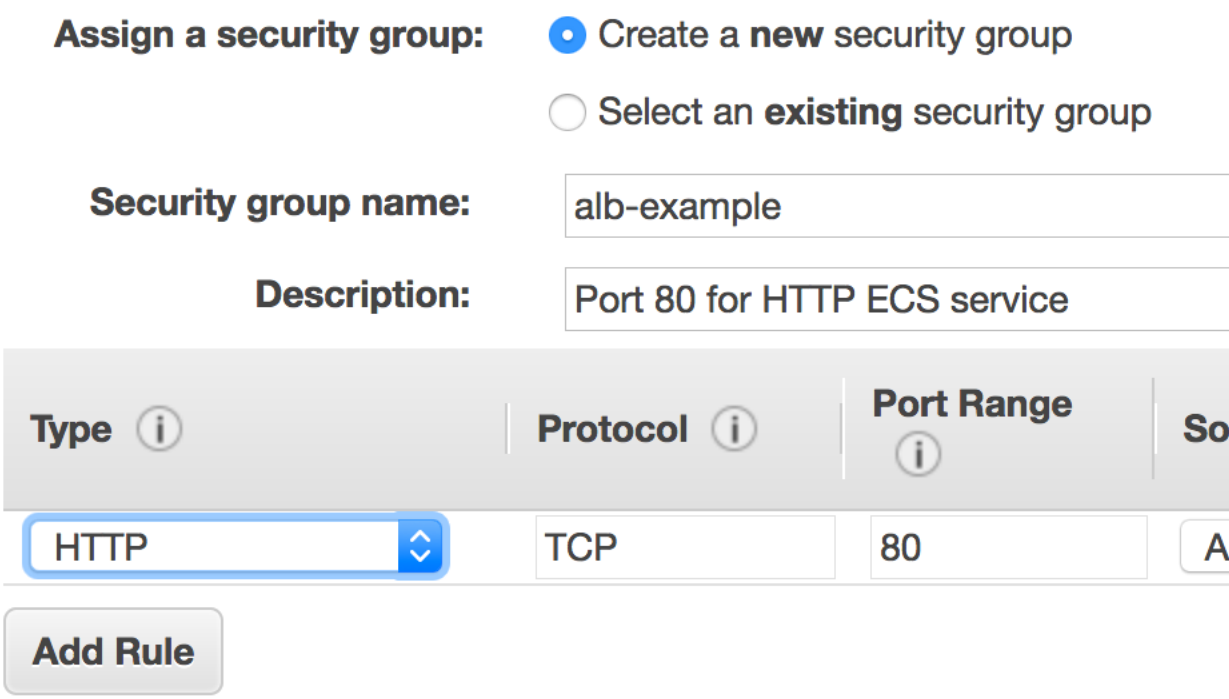
您必须为负载均衡器分配一个安全组，该安全组允许入站流量进入您为侦听器指定的端口。Amazon ECS 不会自动更新与 Elastic Load Balancing 负载均衡器或 Amazon ECS 容器实例关联的安全组。

为您的负载均衡器指定安全组

1. 在 Assign Security Groups 页面上，选择 Create a new security group。
2. 为您的安全组输入名称和描述，或保留默认名称和描述。此新安全组包含一个规则，该规则允许流量进入为要使用的侦听器配置的端口。

Note

在本主题的后面部分，您将为容器实例创建一个安全组规则，该规则允许来自此处创建的安全组的流量进入所有端口，以便 应用程序负载均衡器 可将流量路由到容器实例上的动态分配的主机端口。



3. 选择 Next: Configure Routing 转到向导中的下一页。

配置路由

在此部分中，您将为负载均衡器创建一个目标组，并为已注册到该组的目标创建运行状况检查标准。

创建目标组和配置运行状况检查

1. 对于 Target group，保留默认值 New target group。
2. 对于 Name，键入新目标组的名称。
3. 根据需要设置 Protocol 和 Port。
4. 对于 Target type (目标类型)，选择是否使用实例 ID 或 IP 地址注册您的目标。

   Important

   如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则必须选择 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。
5. 对于 Health checks，保留默认运行状况检查设置。
6. 选择 Next: Register Targets。

注册目标

您的负载均衡器会在已注册到其目标组的目标之间分配流量。将目标组关联到 Amazon ECS 服务时，Amazon ECS 会自动向目标组注册和取消注册容器。由于 Amazon ECS 会处理目标注册，因此您此时不要将目标添加到目标组。

跳过目标注册

1. 在 Registered instances 部分，确保没有选择要注册的示例。
2. 选择 Next: Review 转到向导中的下一页。

审核和创建

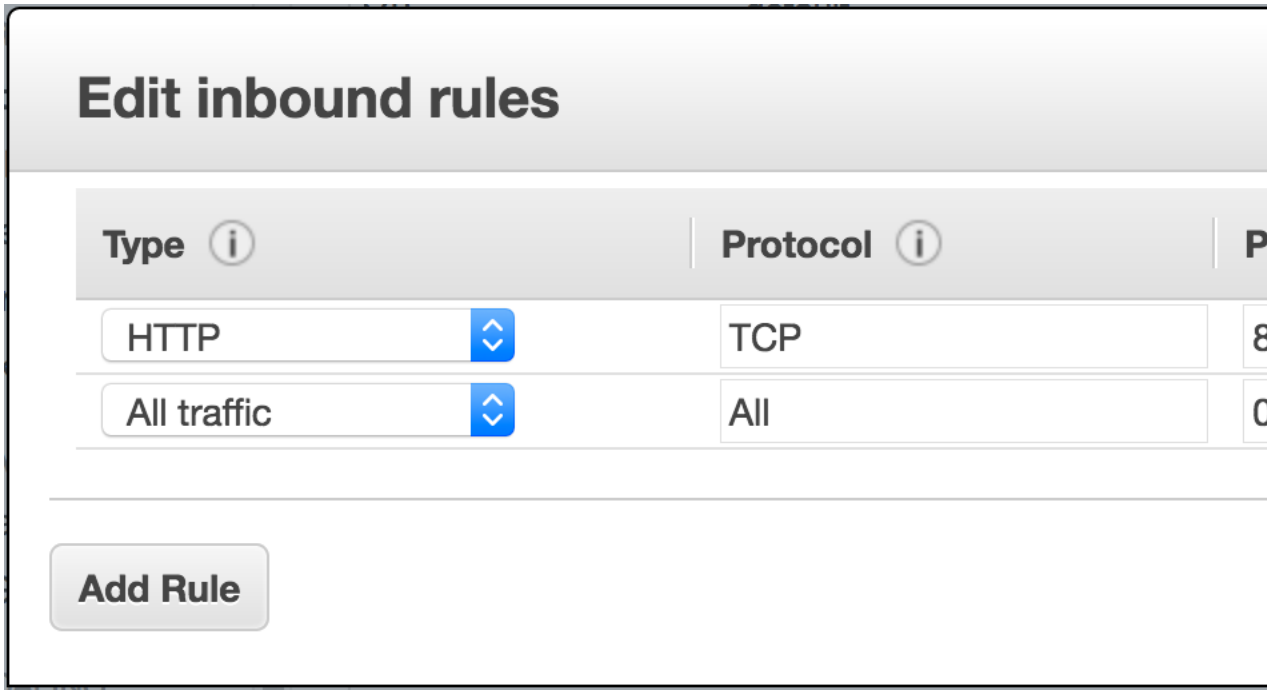
审核您的负载均衡器和目标组配置，并选择 Create 以创建负载均衡器。

为您的容器实例创建安全组规则

创建 应用程序负载均衡器 后，您必须向容器实例安全组添加一个入站规则，该规则允许来自负载均衡器的流量到达容器。

允许来自负载均衡器的入站流量进入容器实例

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格中，选择 Security Groups。
3. 选择您的容器实例使用的安全组。如果已使用 Amazon ECS 首次运行向导创建容器实例，则此安全组可有描述 ECS Allowed Ports (ECS 允许的端口)。
4. 选择 Inbound 选项卡，然后选择 Edit。
5. 对于 Type，选择 All traffic。
6. 对于 Source (源)，选择 Custom (自定义)，然后键入在配置安全组 (p. 290)中创建的应用程序负载均衡器安全组的名称。此规则允许来自 应用程序负载均衡器 的所有流量到达您向负载均衡器注册的任务中的容器。



7. 选择 Save 以完成操作。

创建 Amazon ECS 服务

创建负载均衡器和目标组后，您可在创建服务时在服务定义中指定目标组。在启动服务的每个任务时，服务定义中指定的容器和端口组合将注册到目标组，并且负载均衡器中的流量将路由到该容器。有关更多信息，请参阅创建服务 (p. 317)。

创建 网络负载均衡器

本节指导您完成在 AWS 管理控制台中创建网络负载均衡器的过程。

定义您的负载均衡器

首先，为负载均衡器提供一些基本配置信息（如名称、网络和侦听器）。

侦听器是用于检查连接请求的进程。使用前端 (客户端到负载均衡器) 连接的协议和端口与后端 (负载均衡器到后端实例) 连接的协议和端口配置侦听器。在此实例中，您配置选定网络 (包含接收端口 80 上的 TCP 流量的侦听器) 上面向 Internet 的负载均衡器。

定义您的负载均衡器

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中选择您的负载均衡器的区域。请确保选择您为 Amazon ECS 容器实例选择的同一个区域。
3. 在导航窗格中的 LOAD BALANCING 下，选择 Load Balancers。
4. 选择 Create Load Balancer。
5. 在 Select load balancer type (选择负载均衡器类型) 页面上，选择 网络负载均衡器 下的 Create (创建)。
6. 完成 Configure Load Balancer 页面，如下所示：

   a. 对于 Name，键入负载均衡器的名称。
   b. 对于 Scheme，选择 internet-facing 或 internal。面向 Internet 的负载均衡器将来自客户端的请求通过 Internet 路由到目标。内部负载均衡器使用私有 IP 地址将请求路由到目标。
   c. 对于 Listeners，默认值是负责接收端口 80 上的 TCP 流量的侦听器。您可以保留默认侦听器设置，修改侦听器的协议或端口，或者选择 Add listener 以添加另一个侦听器。

      Note

      如果您计划将流量路由至多个目标组，请参阅 ListenerRules 以获取有关如何添加基于主机或路径的规则的详细信息。

   d. 对于 Availability Zones (可用区)，选择用于 Amazon EC2 实例的 VPC。对于用于启动 Amazon EC2 实例的每个可用区，选择一个可用区，然后为该可用区选择公有子网。要将弹性 IP 地址与子网关联，请从 Elastic IP 选择一个地址。
   e. 选择 Next: Configure Routing。

配置路由

将目标 (如 Amazon EC2 实例) 注册到目标组。您在此步骤中配置的目标组将用作侦听器规则中的目标组，侦听器规则负责将请求转发到目标组。有关更多信息，请参阅您的网络负载均衡器的目标组。

配置目标组

1. 对于 Target group，保留默认值 New target group。
2. 对于 Name，键入目标组的名称。
3. 根据需要设置 Protocol 和 Port。

4. 对于 Target type (目标类型)，选择是否使用实例 ID 或 IP 地址注册您的目标。

   Important

   如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则必须选择 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。
   如果实例具有以下实例类型，则不能用实例 ID 注册实例：
   C1、CC1、CC2、CG1、CG2、CR1、G1、G2、HI1、HS1、M1、M2、M3 和 T1。可以用 IP 地址注册这些类型的实例。
5. 对于 Health checks，保留默认运行状况检查设置。
6. 选择 Next: Register Targets。

将目标注册到目标组

您的负载均衡器会在已注册到其目标组的目标之间分配流量。将目标组关联到 Amazon ECS 服务时，Amazon ECS 会自动向目标组注册和取消注册容器。由于 Amazon ECS 会处理目标注册，因此您此时不要将目标添加到目标组。

跳过目标注册

1. 在 Registered instances 部分，确保没有选择要注册的示例。
2. 选择 Next: Review 转到向导中的下一页。

审核和创建

审核您的负载均衡器和目标组配置，并选择 Create 以创建负载均衡器。

创建 Amazon ECS 服务

创建负载均衡器和目标组后，您可在创建服务时在服务定义中指定目标组。在启动服务的每个任务时，服务定义中指定的容器和端口组合将注册到目标组，并且负载均衡器中的流量将路由到该容器。有关更多信息，请参阅创建服务 (p. 317)。

创建 传统负载均衡器

本节指导您完成在 AWS 管理控制台中创建传统负载均衡器的过程。

您可以创建与 EC2-Classic 或 VPC 结合使用的 传统负载均衡器。这些过程中描述的某些任务仅适用于 VPC 中的负载均衡器。

定义您的负载均衡器

首先，为负载均衡器提供一些基本配置信息（如名称、网络和侦听器）。

侦听器是用于检查连接请求的进程。使用前端 (客户端到负载均衡器) 连接的协议和端口与后端 (负载均衡器到后端实例) 连接的协议和端口配置侦听器。在此示例中，您将配置一个侦听器，该侦听器接受端口 80 上的 HTTP 请求并使用 HTTP 将它们发送至端口 80 上的后端实例。

定义您的负载均衡器

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中选择您的负载均衡器的区域。请确保选择您为 Amazon ECS 容器实例选择的同一个区域。
3. 在导航窗格中的 LOAD BALANCING 下，选择 Load Balancers。
4. 选择 Create Load Balancer。
5. 在 Select load balancer type (选择负载均衡器类型) 页面上，选择 传统负载均衡器。

6. 对于 Load Balancer name，输入负载均衡器的唯一名称。

   您选择的负载均衡器名称必须在负载均衡器集中唯一，必须最多包含 32 个字符，并且必须仅包含字母数字或连字符。
7. 对于 Create LB inside，选择您的容器实例所在的相同网络：EC2-Classic 或特定 VPC。
8. 默认值配置一个 HTTP 负载均衡器，该负载均衡器将负载均衡器上端口 80 中的流量转发到容器实例的端口 80，但您可以更改应用程序的这些值。有关更多信息，请参阅 Classic Load Balancer 用户指南 中的 Classic Load Balancer 的侦听器。
9. [EC2-VPC] 要提高负载均衡器的可用性，请至少选择不同可用区中的两个子网。您的负载均衡器子网配置必须包含容器实例所在的所有可用区。在 Select Subnets 部分中的 Available Subnets 下，选择子网。您选择的子网将移至 Selected Subnets 下。

   Note

   如果您选择 EC2-Classic 作为您的网络，或者您拥有默认 VPC 但没有选择 Enable advanced VPC configuration，您将看不到 Select Subnets。

Available Subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
⊕	us-west-2c	subnet-cb663da2	10.0.1.0/24	
⊕	us-west-2c	subnet-c9663da0	10.0.0.0/24	

Selected Subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
⊖	us-west-2a	subnet-e4f33493	10.0.2.0/24	
⊖	us-west-2b	subnet-5264e837	10.0.3.0/24	

10. 选择 Next: Assign Security Groups 以转到向导中的下一页。

为 VPC 中的负载均衡器分配安全组

如果在 VPC 中创建了负载均衡器，则必须为其分配一个安全组，该安全组允许入站流量进入为负载均衡器指定的端口以及针对负载均衡器的运行状况检查。Amazon ECS 不会自动更新与 Elastic Load Balancing 负载均衡器或 Amazon ECS 容器实例关联的安全组。

Note

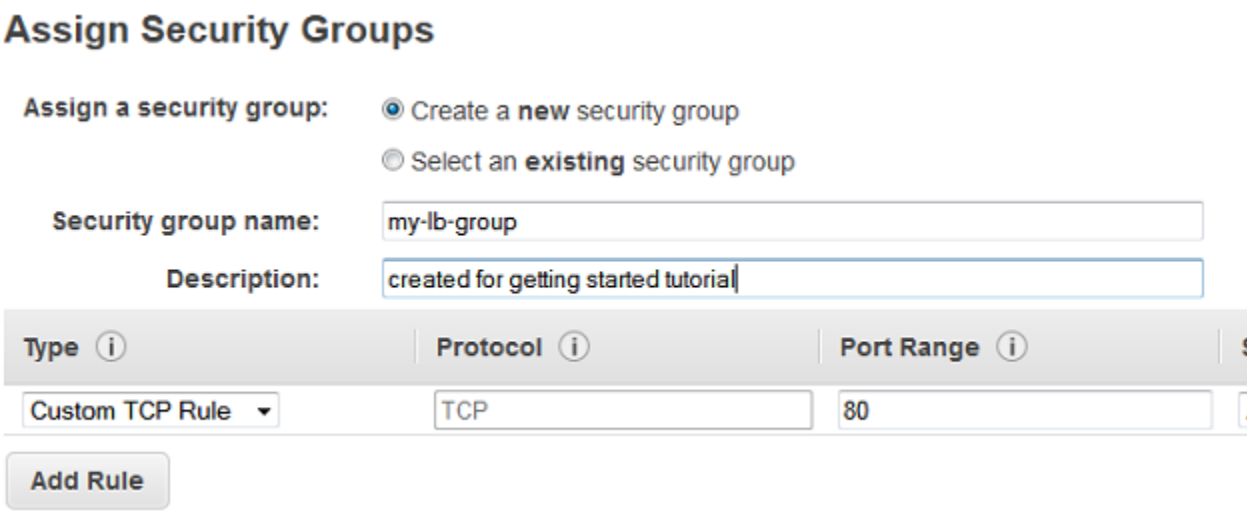
如果选择 EC2-Classic 作的网络，则向导中不会显示此页面，您可以转到下一步。Elastic Load Balancing 提供了一个安全组，该安全组自动分配到 EC2-Classic 的负载均衡器。

为您的负载均衡器指定安全组

1. 在 Assign Security Groups 页面上，选择 Create a new security group。
2. 为您的安全组输入名称和描述，或保留默认名称和描述。此新安全组包含一个规则，该规则允许流量进入为要使用的负载均衡器配置的端口。如果您为运行状况检查指定了其他端口，则必须选择 Add Rule 来添加一个规则以允许入站流量也进入该端口。

   Note

   还应将此安全组分配到服务中的容器实例或具有相同规则的另一个安全组。



3. 选择 Next: Configure Security Settings 以转到向导中的下一页。

（可选）配置安全设置

在本教程中，您可以选择 Next: Configure Health Check 继续执行下一步骤。有关创建 HTTPS 负载均衡器和使用其他安全功能的更多信息，请参阅 Classic Load Balancer 用户指南 中的 HTTPS 负载均衡器。

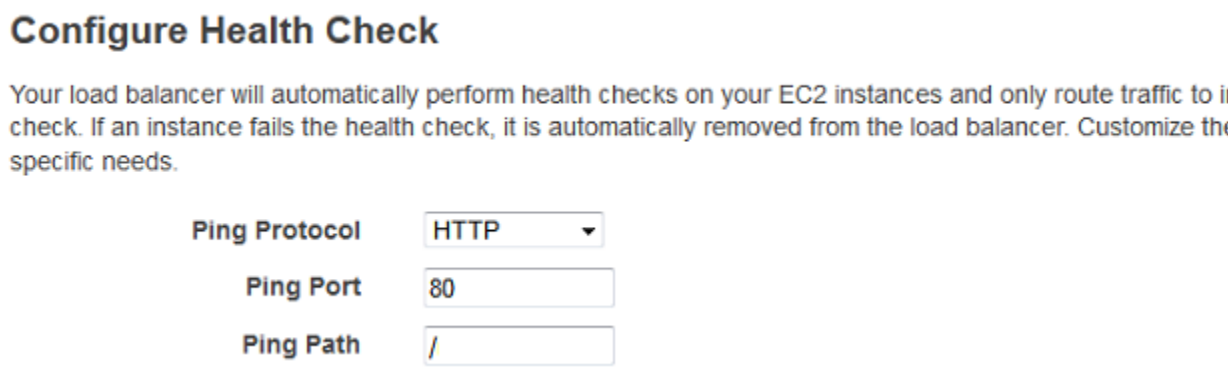
为您的 EC2 实例配置运行状况检查

Elastic Load Balancing 将自动检查服务中的任务的运行状况。如果 Elastic Load Balancing 发现某一运行不正常的任务，它会停止向该实例发送流量，并重新路由以将流量发送至正常运行的实例。Amazon ECS 将停止运行不正常的任务并启动该任务的其他实例。

> **Note**
>
> 以下过程将配置一个 HTTP（端口 80）负载均衡器，但您可以为应用程序修改这些值。

为您的实例配置运行状况检查

1. 在 Configure Health Check (配置运行状况检查) 页面上，执行以下操作：
   a. 将 Ping Protocol 设置为其默认值 `HTTP`。
   b. 将 Ping Port 设置为其默认值 `80`。
   c. 对于 Ping Path，将默认值替换为单斜线 (“/”)。这会指示 Elastic Load Balancing 将运行状况检查查询发送到 Web 服务器的默认主页，如 `index.html` 或 `default.html`。
   d. 将其他字段保留为默认值。



2. 选择 Next: Add EC2 Instances 以转到向导中的下一页。

负载均衡器实例注册

负载均衡器会在向其注册的实例间分配流量。将负载均衡器分配到 Amazon ECS 服务时，Amazon ECS 会在服务中的任务在其上运行时注册和取消注册容器实例。由于 Amazon ECS 处理容器实例注册，因此您此时不要将容器实例添加到负载均衡器。

跳过实例注册并为负载均衡器添加标签

1. 在 Add EC2 Instances 页面上，对于 Add Instances to Load Balancer，确保没有选择要注册的实例。
2. 将其他字段保留为默认值。
3. 选择 Next: Add Tags 以转到向导中的下一页。

为您的负载均衡器添加标签（可选）

您可以为您的负载均衡器添加标签，或者继续进行下一个步骤。您可以稍后为负载均衡器添加标签。有关更多信息，请参阅 Classic Load Balancer 用户指南 中的为 Classic Load Balancer 添加标签。

向负载均衡器添加标签

1. 在 Add Tags (添加标签) 页面，为标签指定键和值。
2. 要添加其他标签，请选择 Create Tag 并为标签指定键和值。
3. 添加完标签后，选择 Review and Create。

创建并验证您的负载均衡器

在创建负载均衡器之前，请检查所选的设置。创建负载均衡器后，您可以创建一个服务，该服务使用负载均衡器验证其是否正在将流量发送到容器实例。

完成对负载均衡器的创建

1. 在 Review (审核) 页面上，检查您的设置。要更改初始设置，请选择相应的编辑链接。
2. 选择 Create 以创建负载均衡器。
3. 在您收到已创建负载均衡器的通知后，选择 Close。

创建 Amazon ECS 服务

创建负载均衡器后，您可以在创建服务时在服务定义中指定它。有关更多信息，请参阅 创建服务 (p. 317)。

向服务注册多个目标组

当您在服务定义中指定多个目标组时，Amazon ECS 服务可以为来自多个负载平衡器的流量提供服务并公开多个负载平衡端口。

目前，如果要创建一个指定多个目标组的服务，您必须使用 Amazon ECS API、开发工具包、AWS CLI 或 AWS CloudFormation 模板创建该服务。在创建服务后，您可以使用 AWS 管理控制台查看服务以及注册到服务的目标组。

可以使用以下格式在服务定义中指定多个目标组。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"container_name",
      "containerPort":container_port
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"container_name",
      "containerPort":container_port
   }
]
```

多个目标组的注意事项

在服务定义中指定多个目标组时，应注意以下事项：

- 仅在使用 应用程序负载均衡器 或 网络负载均衡器 负载均衡器类型时支持多个目标组。
- 仅在服务使用滚动更新 (ECS) 部署控制器类型时支持多个目标组。如果您使用 CodeDeploy 或外部部署控制器，则不支持多个目标组。
- 包含使用 Fargate 和 EC2 启动类型的任务的服务支持多个目标组。

示例服务定义

下面是一些在服务定义中指定多个目标组的示例使用案例。

示例：为内部和外部流量使用独立的负载均衡器

在以下使用案例中，服务对相同的容器和端口使用两个独立的负载均衡器，一个用于内部流量，另一个用于面向 Internet 的流量。

```
"loadBalancers":[
   //Internal ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"nginx",
```

```
      "containerPort":8080
   },
   //Internet-facing ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"nginx",
      "containerPort":8080
   }
]
```

示例：从同一容器公开多个端口

在以下使用案例中，服务使用一个负载均衡器，但从同一容器公开多个端口。例如，Jenkins 容器可能会为 Jenkins Web 接口公开端口 8080，为 API 公开端口 50000。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"jenkins",
      "containerPort":8080
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"jenkins",
      "containerPort":50000
   }
]
```

示例：从多个容器公开端口

在以下使用案例中，服务使用一个负载均衡器和两个目标组从单独的容器公开端口。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"webserver",
      "containerPort":80
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"database",
      "containerPort":3306
   }
]
```

服务 Auto Scaling

您可以选择将 Amazon ECS 服务配置为使用 服务 Auto Scaling 以自动上调或下调其预期数量。服务 Auto Scaling 利用 Application Auto Scaling 服务来提供此功能。服务 Auto Scaling 在支持 Amazon ECS 的所有区域中可用。有关更多信息，请参阅《Application Auto Scaling 用户指南》。

Amazon ECS 服务 Auto Scaling 支持以下类型的扩展策略：

- 目标跟踪扩展策略 (p. 300)—根据特定 CloudWatch 指标的目标值，增加或减少服务运行的任务数。这与恒温器保持家里温度的方式类似。您选择一个温度，恒温器将完成所有其他工作。
- 步进扩展策略 (p. 305)—增加或减少服务为响应 CloudWatch 警报而运行的任务数。步进扩展基于一组扩展调整（称作步进调整），这些调整因警报违例大小而异。

服务 Auto Scaling 需要 IAM 权限

可将 Amazon ECS、CloudWatch 和 Application Auto Scaling API 组合使用以提供服务 Auto Scaling。服务是使用 Amazon ECS 创建和更新的，警报是使用 CloudWatch 创建的，扩展策略是使用 Application Auto Scaling 创建的。IAM 用户必须先拥有这些服务的适当权限，然后才能在 AWS 管理控制台、AWS CLI 或开发工具包中与扩展策略进行交互。除了用于创建和更新服务的标准 IAM 权限之外，服务 Auto Scaling 还需要以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:*",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

创建服务示例 (p. 383)和更新服务示例 (p. 384) IAM 策略示例说明 IAM 用户在 AWS 管理控制台中使用服务 Auto Scaling 所需的权限。

Application Auto Scaling 服务需要描述 Amazon ECS 服务和 CloudWatch 警报的权限，以及代表您修改服务的预期数量的权限。您必须为 ECS 服务创建 IAM 角色 (`ecsAutoscaleRole`) 以提供这些权限，然后将该角色与服务关联，以便它才能使用 Application Auto Scaling。如果一个 IAM 用户有权在 Amazon ECS 控制台中使用服务 Auto Scaling，创建 IAM 角色，并将 IAM 角色策略附加到这些角色，则该用户可自动创建此角色（包含在 Amazon ECS 控制台创建服务 (p.)或更新服务 (p. 326)工作流程中），然后将此角色用于任何其他服务（在控制台中或通过 CLI 或开发工具包）。您也可以使用 Amazon ECS 服务 Auto Scaling IAM 角色 (p. 414)中的过程来创建角色。

或者，Application Auto Scaling 也可以利用服务相关角色 (`AWSServiceRoleForApplicationAutoScaling_ECSService`)，该角色代表您执行自动扩展操作。如果您具有所需的权限，则当您开始使用 Application Auto Scaling 服务时，将自动为您创建此角色并附加所需的 IAM 策略。有关更多信息，请参阅 Application Auto Scaling 用户指南 中的用于 Application Auto Scaling 的服务相关角色。您或管理员必须具有 `iam:CreateServiceLinkedRole` 或 `AdministratorAccess` 权限才能创建此服务相关角色。

目标跟踪扩展策略

在使用目标跟踪扩展策略时，可以选择一个 CloudWatch 指标并设置一个目标值。Amazon ECS 创建和管理触发扩展策略的 CloudWatch 警报，并根据指标和目标值计算扩展调整。扩展策略根据需要增加或删除服务任务，将指标保持在指定的目标值或接近指定的目标值。除了将指标保持在目标值附近以外，目标跟踪扩展策略还会根据由于负载模式波动而造成的指标波动进行调节，并最大限度减少服务中运行的任务数发生快速波动的情况。

您可以为 Amazon ECS 服务创建多个目标跟踪扩展策略，但前提是它们分别使用不同的指标。服务根据提供最大任务容量的策略进行扩展。这样，您就可以涵盖多种方案，并确保始终具有足够的容量以处理您的应用程序工作负载。

为了确保应用程序可用性，服务会针对指标尽快按比例扩展，但缩减过程相对缓慢。

请勿编辑或删除 Amazon ECS 为目标跟踪扩展策略管理的 CloudWatch 警报。在删除目标跟踪扩展策略时，Amazon ECS 将自动删除警报。

注意事项

创建目标跟踪扩展策略时，请注意以下注意事项：

- 目标跟踪扩展策略假设它应该在指定指标高于目标值时执行向外扩展。因此，不能使用目标跟踪扩展策略在指定指标低于目标值时向外扩展。
- 当指定指标数据不足时，目标跟踪扩展策略不会执行扩展。它不会执行向内扩展，因为它不会将数据不足解读为使用率低。
- 您可能会看到目标值与实际指标数据点之间存在差距。这是因为 Application Auto Scaling 在确定要添加或删除多少容量时将始终通过向上或向下舍入保守地进行操作，以免添加的容量不足或删除的容量过多。但是，对于具有小容量的可扩展目标，实际指标数据点可能与目标值差距很大。对于容量更高的可扩展目标，添加或删除容量将缩小目标值和实际指标数据点之间的差距。
- 建议将基于指标扩展的频率设置为 1 分钟，这可更快地响应使用率变化。如果将随指标扩展的频率设置为 5 分钟，可能会导致响应时间变慢，并且可能导致系统依据陈旧的指标数据进行扩展。
- 为了确保应用程序可用性，Application Auto Scaling 针对指标尽快按比例向外扩展，但会逐渐向内扩展。
- 请勿编辑或删除 Application Auto Scaling 为目标跟踪扩展策略管理的 CloudWatch 警报。在删除扩展策略时，Application Auto Scaling 将自动删除警报。

教程：使用目标跟踪的服务自动扩展

以下过程帮助您创建一个 Amazon ECS 集群和一种通过目标跟踪使用 Application Auto Scaling 进行扩展（和缩减）的服务。

Amazon ECS 发布 CloudWatch 指标与服务的平均 CPU 和内存使用率。您可以使用这些服务利用率指标扩展服务以满足高峰期的大量需求，也可以缩减服务以在低利用率期间降低成本。有关更多信息，请参阅 服务利用率 (p. 347)。

在本教程中，将使用 Amazon ECS 首次运行向导创建一个集群和一个服务（在 Elastic Load Balancing 负载均衡器后面运行）。然后，使用 CloudWatch 警报在服务上配置服务 Auto Scaling，该警报使用 `ECSServiceAverageCPUUtilization` 指标扩展或缩减服务，具体取决于当前应用程序负载。

当您的服务的 CPU 利用率上升至 75% 以上 (这意味着，已使用为服务预留的 CPU 的 75% 以上) 时，扩展警报将触发 服务 Auto Scaling 以向您的服务添加另一个任务来帮助处理增加的负载。相反，当您的服务的 CPU 利用率下降至 75% 以下时，向内扩展警报将促使服务预期数量减少，以便为其他资源和服务释放集群资源。

先决条件

本教程假定您有一个 AWS 账户和一个有权执行教程中介绍的所有操作的 IAM 管理员，以及一个位于当前区域的 Amazon EC2 密钥对。如果您没有这些资源（或不确定是否有），则可以执行 Amazon ECS 的设置 (p. 7)中的步骤来创建这些资源。

如果在 CloudWatch 指标可用于 Amazon ECS 之前已创建 Amazon ECS 容器实例，则可能需要添加此权限。有关检查 Amazon ECS 容器实例角色并为容器实例附加托管 IAM 策略的信息，请参阅在 IAM 控制台中检查 `ecsInstanceRole` (p. 405)。

步骤 1：创建集群和服务

为集群和服务启用 CloudWatch 指标后，可以使用 Amazon ECS 首次运行向导创建集群和服务。首次运行向导负责为本教程创建所需的 IAM 角色和策略（一个用于容器实例的 Auto Scaling 组），它还创建在负载均衡器后面运行的服务。该向导还将使清理过程变得轻松得多，因为您可以通过一个步骤删除整个 AWS CloudFormation 堆栈。

对于本教程，您将创建一个名为 `service-autoscaling` 的集群和一个名为 `sample-webapp` 的服务。

创建集群和服务

1. 打开 https://console.aws.amazon.com/ecs/home#/firstRun 上的 Amazon ECS 控制台首次运行向导。
2. 从导航栏中，选择美国东部 (弗吉尼亚北部) 区域。
3. 在 Step 1: Container and Task (步骤 1: 容器和任务) 中，对于 Container definition (容器定义) 选择 sample-app (示例应用)。
4. 对于 Task definition (任务定义)，保留所有默认选项并选择 Next (下一步)。
5. 在 Step 2: Service (步骤 2: 服务) 中，对于 Load balancer type (负载均衡器类型) 选择 Application Load Balancer (应用程序负载均衡器)、Next (下一步)。

   > **Important**
   >
   > AWS 资源中有应用程序负载均衡器时，将产生相应费用。有关更多信息，请参阅 Elastic Load Balancing 定价。

6. 在 Step 3: Cluster (步骤 3: 集群) 中，对于 Cluster name (集群名称) 输入 `service-autoscaling` 并选择 Next (下一步)。
7. 检查您的选择，然后选择 Create (创建)。

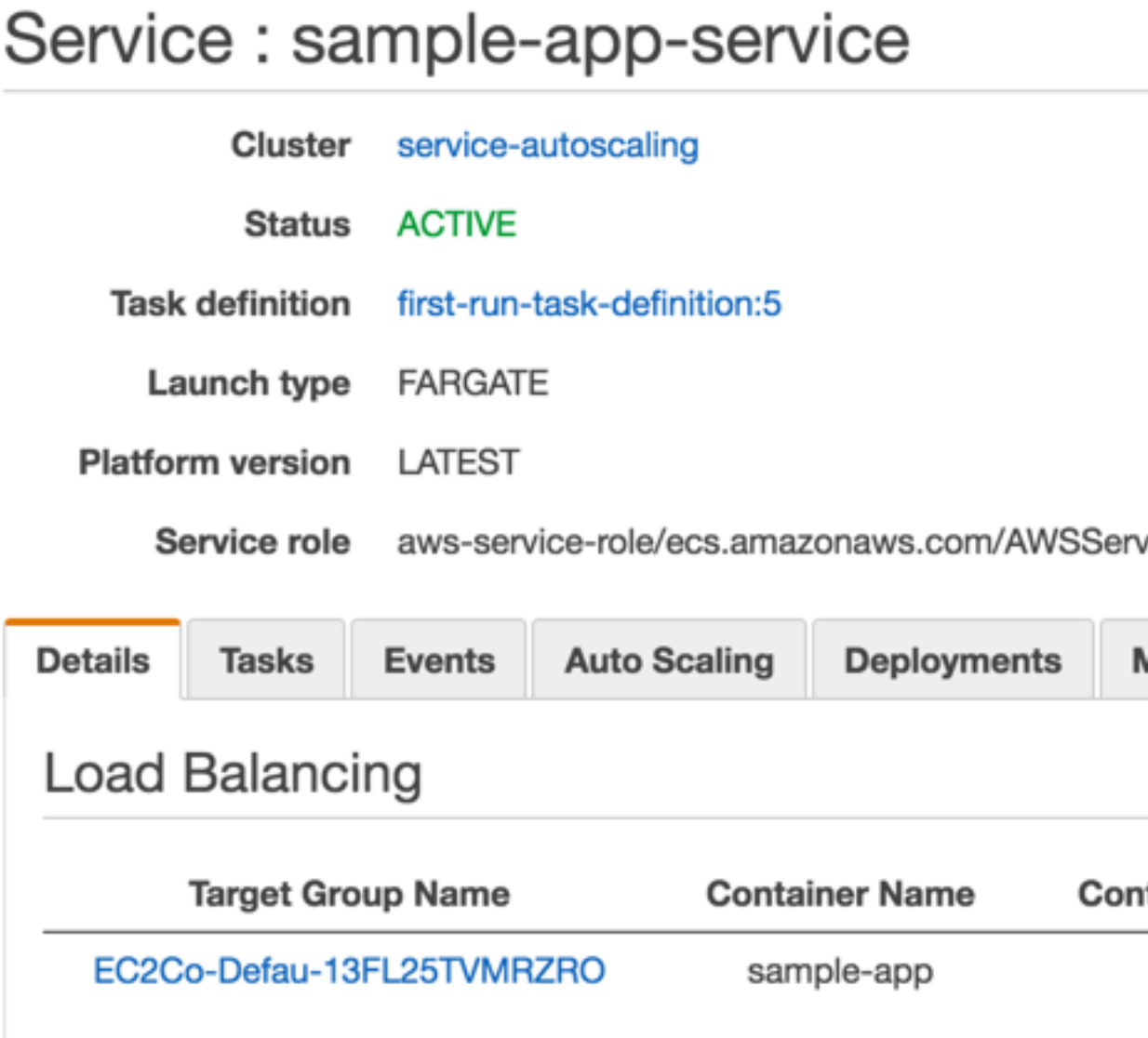
   您将会转到 Launch Status (启动状态) 页面，此页面显示您的启动状态并描述此过程的每一步 (在创建和填充您的集群资源时，这可能需要几分钟时间才能完成)。
8. 创建集群和服务后，选择 View service (查看服务)。

步骤 2：配置 服务 Auto Scaling

现在，您已启动一个集群，并且已在该集群中创建一个在负载均衡器后面运行的服务，您可以通过创建根据 CloudWatch 警报扩展和缩减服务的扩展策略来配置服务 Auto Scaling。

配置基本 服务 Auto Scaling 参数

1. 在 Service: sample-app-service (服务: sample-app-service) 页面上，您的服务配置应类似于下图（尽管任务定义修订和负载均衡器名称可能会不同）。选择 Update 以更新您的新服务。



2. 在 Update service (更新服务) 页面上选择 Next step (下一步)，直至到达 Step 3: Set Auto Scaling (optional) (步骤 3: 设置 Auto Scaling (可选))。
3. 对于 服务 Auto Scaling，选择 Configure Service Auto Scaling to adjust your service's desired count (配置服务 Auto Scaling 以调整服务的预期数量)。
4. 对于 Minimum number of tasks (最小任务数)，输入 `1` 作为供服务 Auto Scaling 使用的任务数的下限。您的服务的预期数量不会自动调整到低于此数量。
5. 对于 Desired number of tasks (预期任务数)，此字段是使用您之前输入的值预先填充的。此值必须介于此页面上指定的最小任务数和最大任务数之间。将此值保留为 `1`.
6. 对于 Maximum number of tasks (最大任务数)，输入 `2` 作为供服务 Auto Scaling 使用的任务数的上限。您的服务的预期数量不会自动调整到高于此数量。
7. 对于 IAM role for Service Auto Scaling (与服务 Auto Scaling 配合使用的 IAM 角色)，选择一个 IAM 角色来授权 Application Auto Scaling 服务代表您调整服务的预期数量。如果您之前未创建此类角色，请选择 Create new role，系统将为您创建此角色。为了方便将来参考，为您创建的角色名为 `ecsAutoscaleRole`。有关更多信息，请参阅 Amazon ECS 服务 Auto Scaling IAM 角色 (p. 414)。

为您的服务配置扩展策略

以下步骤将帮助您创建扩展策略和可用于为您的服务触发扩展活动的 CloudWatch 警报。您可以创建扩展警报来增加服务的预期数量，并创建缩减警报来减少服务的预期数量。

1. 选择 Add scaling policy (添加扩展策略) 以配置您的扩展策略。
2. 在 Add policy (添加策略) 页面上，更新以下字段：
   a. 对于 Scaling policy type (扩展策略类型) 选择 Target tracking (目标跟踪)。
   b. 对于 Policy name，输入 `TargetTrackingPolicy`.
   c. 对于 ECS service metric (ECS 服务指标)，选择 ECSServiceAverageCPUUtilization。
   d. 对于 Target value (目标值) 输入 `75`。
   e. 对于 Scale-out cooldown period (扩展冷却时间)，输入 `60`。这是在一个扩展活动完成后、另一个扩展活动开始前的时长 (秒)。在此期间，已启动的资源不影响 Auto Scaling 组指标。
   f. 对于 Scale-in cooldown period (缩减冷却时间)，输入 `60`。这是一个缩减活动完成后、另一个缩减活动开始前的时长 (秒)。在此期间，已启动的资源不影响 Auto Scaling 组指标。
   g. 选择 Save。
3. 选择下一步。
4. 检查您的所有选择，然后选择 Update Service (更新服务)。
5. 当您的服务状态为已完成更新时，选择 View Service。

步骤 3：触发扩展活动

使用服务 Auto Scaling 配置服务后，可以通过将服务的 CPU 利用率推送到 `ALARM` 状态来触发扩展活动。由于本教程中的示例是一个在负载均衡器后面运行的 Web 应用程序，因此，您可以将数千个 HTTP 请求发送到服务 (使用 ApacheBench 实用工具) 以使服务 CPU 利用率迅速上升并超过阈值。此峰值将触发警报，而警报反过来会触发一个扩展活动来向服务添加一个任务。

ApacheBench 实用工具完成请求后，服务 CPU 利用率应降至 25% 阈值以下，并触发将服务的预期数量返回至 1 的向内扩展活动。

为服务触发扩展活动

1. 从控制台中的服务主视图页面，选择负载均衡器名称以在 Amazon EC2 控制台中查看其详细信息。您需要负载均衡器的 DNS 名称，类似于如下内容：`EC2Contai-EcsElast-SMAKV74U23PH-96652279.us-east-1.elb.amazonaws.com`。
2. 使用 ApacheBench (ab) 实用工具在短时间内向负载均衡器发出数千个 HTTP 请求。

   Note

   此命令在 macOS 上是默认安装的，也可用于许多 Linux 分发版。例如，您可以在 Amazon Linux 上通过以下命令安装 ab：

   ```
   $ sudo yum install -y httpd24-tools
   ```

   运行以下命令，将替换负载均衡器的 DNS 名称。

   ```
   $ ab -n 100000 -c 1000 http://EC2Contai-EcsElast-SMAKV74U23PH-96652279.us-east-1.elb.amazonaws.com/
   ```

3. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
4. 在左侧导航窗格中，选择 Alarms (警报)。
5. 等待 ab HTTP 请求在 CloudWatch 控制台中触发向外扩展警报。您应该会看到您的 Amazon ECS 服务扩展并将 1 个任务添加到服务的预期数量。

6. 在 ab HTTP 请求完成后的短时间内（1 到 2 分钟）缩减警报将触发，缩减策略将服务的预期数量减至 1。

步骤 4：清理

完成本教程后，您可以选择保留您的集群、Auto Scaling 组、负载均衡器和 CloudWatch 警报。但是，如果您没有主动使用这些资源，则应考虑将其清除以免您的账户产生不必要的费用。

删除集群和 CloudWatch 警报

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在左侧导航窗格中，选择集群。
3. 在 Clusters (集群) 页面上，选择 service-autoscaling (service-autoscaling) 集群。
4. 选择 Delete Cluster (删除集群)、Delete (删除)。集群 AWS CloudFormation 堆栈的清理可能需要几分钟才能完成。
5. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
6. 选择 Alarms (警报) 并选择以 `TargetTracking-service` 开头的警报。
7. 选择 Delete (删除)、Yes, Delete (是，删除)。

步进扩展策略

在使用步进扩展策略时，您可以指定 CloudWatch 警报以触发扩展过程。例如，如果您希望在 CPU 利用率达到特定水平时扩展，可以使用 Amazon ECS 提供的 `CPUUtilization` 指标创建警报。

Amazon ECS 发布 CloudWatch 指标与服务的平均 CPU 和内存使用率。您可以使用这些服务利用率指标扩展服务以满足高峰期的大量需求，也可以缩减服务以在低利用率期间降低成本。有关更多信息，请参阅服务利用率 (p. 347)。

对于包含使用 EC2 启动类型的任务的服务，您可以使用 CloudWatch 警报扩展容器实例。有关更多信息，请参阅 教程：通过 CloudWatch 警报扩展容器实例 (p. 349)。您也可以使用由其他服务发布的 CloudWatch 指标或特定于您的应用程序的自定义指标。例如，Web 服务可增加基于 Elastic Load Balancing 指标的任务（例如 `SurgeQueueLength`）的数目，批处理作业可增加基于 Amazon SQS 指标的任务（例如 `ApproximateNumberOfMessagesVisible`）的数目。

步进扩展概念

- ECS 服务计划程序始终遵守预期数量，但只要您的服务拥有活动的扩展策略和警报，服务 Auto Scaling 就会更改您手动设置的预期数量。
- 如果设置的服务的预期数量低于其最小容量值，并且警报触发了扩展活动，则 Application Auto Scaling 会将预期数量增至最小容量值，然后基于与警报关联的扩展策略继续按需扩展。但是，缩减活动将不会调整预期数量，因为它已低于最小容量值。
- 如果设置的服务的预期数量高于其最大容量值，并且警报触发了向内扩展活动，则 Application Auto Scaling 会将预期数量缩减至到最大容量值，然后基于与警报关联的扩展策略继续按需向内扩展。但是，扩展活动将不会调整预期数量，因为它已高于最大容量值。
- 在扩展活动期间，服务中实际运行的任务数是 服务 Auto Scaling 用作其起点的值（与预期数量相对），它是应有的处理容量。这可阻止无法满足的过度（失控）扩展，例如，在容器实例资源不足，无法放置其他任务的情况下。如果稍后提供容器实例容量，则正在等待的扩展活动将能够继续，随后其他扩展活动可在冷却期后继续。

Amazon ECS 控制台体验

Amazon ECS 控制台的服务创建和服务更新工作流程支持步进扩展策略。Amazon ECS 控制台处理 `ecsAutoscaleRole` 和策略创建，前提是使用控制台的 IAM 用户具有服务 Auto Scaling 需要 IAM 权限 (p. 300)中描述的权限，并且他们能够创建 IAM 角色并为角色附加策略。

在控制台中将某项服务配置为使用服务 Auto Scaling 时，服务将自动作为可扩展目标注册到 Application Auto Scaling，以便您能够配置扩展和缩减服务的扩展策略。您也可以创建和更新扩展策略以及在 Amazon ECS 控制台中触发这些策略的 CloudWatch 警报。

要创建使用服务 Auto Scaling 的新 ECS 服务，请参阅创建服务 (p. 317)。

要将现有服务更新为使用服务 Auto Scaling，请参阅更新服务 (p. 326)。

AWS CLI 和开发工具包体验

您可以使用 AWS CLI 或 AWS 开发工具包配置服务 Auto Scaling，但必须遵循以下注意事项。

- 可将 Amazon ECS、CloudWatch 和 Application Auto Scaling API 组合使用以提供服务 Auto Scaling。服务是使用 Amazon ECS 创建和更新的，警报是使用 CloudWatch 创建的，扩展策略是使用 Application Auto Scaling 创建的。有关这些特定的 API 操作的更多信息，请参阅 Amazon Elastic Container Service API Reference、Amazon CloudWatch API Reference和 Application Auto Scaling API 参考。有关适用于这些服务的 AWS CLI 命令的更多信息，请参阅 AWS CLI Command Reference的 ecs、cloudwatch 和 application-autoscaling 部分。
- 您必须先将服务作为可扩展目标注册到 Application Auto Scaling RegisterScalableTarget API 操作，然后服务才能使用服务 Auto Scaling。
- 将 ECS 服务注册为可扩展目标后，可以使用 Application Auto Scaling PutScalingPolicy API 操作创建扩展策略以指定触发 CloudWatch 警报时应发生的情况。
- 为服务创建扩展策略后，可以使用 CloudWatch PutMetricAlarm API 操作为服务创建触发扩展事件的 CloudWatch 警报。

服务发现

Amazon ECS 服务可以选择配置为使用 Amazon ECS 服务发现。服务发现使用 AWS Cloud Map API 操作来管理 Amazon ECS 服务的 HTTP 和 DNS 命名空间。有关更多信息，请参阅 AWS Cloud Map 开发人员指南 中的什么是 AWS Cloud Map？。

主题

- 服务发现 概念 (p. 306)
- 服务发现 注意事项 (p. 307)
- Amazon ECS 控制台体验 (p. 308)
- 服务发现 定价 (p. 308)
- 教程：使用 服务发现 创建服务 (p. 308)

服务发现 概念

服务发现 包括以下组件：

- 服务发现命名空间：共享相同域名的服务发现服务的逻辑组，例如 `example.com`。
- 服务发现 服务：存在于 服务发现 命名空间中，由命名空间的服务名称和 DNS 配置组成。它提供了以下核心组件：
  - 服务注册：让您可通过 DNS 或 AWS Cloud Map API 操作查找服务，并获取一个或多个可用于连接到该服务的可用终端节点。

- 服务发现实例：存在于服务发现服务中并包含与服务目录中的每个 Amazon ECS 服务相关联的属性。
  - 实例属性：将以下元数据添加为配置为使用服务发现的每个 Amazon ECS 服务的自定义属性：
    - **AWS_INSTANCE_IPV4** – 对于 A 记录，为 Route 53 响应 DNS 查询而返回和 AWS Cloud Map 在发现实例详细信息时返回的 IPv4 地址，例如 192.0.2.44。
    - **AWS_INSTANCE_PORT** – 与服务发现服务关联的端口值。
    - **AVAILABILITY_ZONE** – 任务启动到的可用区。对于使用 EC2 启动类型的任务，这是容器实例存在于的可用区。对于使用 Fargate 启动类型的任务，这是弹性网络接口存在于的可用区。
    - **REGION** – 任务存在于的区域。
    - **ECS_SERVICE_NAME** – 任务属于的 Amazon ECS 服务的名称。
    - **ECS_CLUSTER_NAME** – 任务属于的 Amazon ECS 集群的名称。
    - **EC2_INSTANCE_ID** – 在其上放置任务的容器实例的 ID。如果任务使用的是 Fargate 启动类型，则不会添加此自定义属性。
    - **ECS_TASK_DEFINITION_FAMILY** – 任务使用的任务定义系列。
    - **ECS_TASK_SET_EXTERNAL_ID** – 如果为外部部署创建任务集并将任务集与服务发现注册表关联，ECS_TASK_SET_EXTERNAL_ID 属性将包含任务集的外部 ID。
- Amazon ECS 运行状况检查：Amazon ECS 执行定期容器级别的运行状况检查。如果终端节点不传递运行状况检查，则会将其从 DNS 路由中删除并标记为不正常。

服务发现 注意事项

使用 服务发现 时应注意以下事项：

- 如果使用平台版本 v1.1.0 或更高版本，则使用 Fargate 启动类型的任务支持服务发现。有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。
- Amazon ECS 控制台中的创建服务工作流程仅支持向私有 DNS 命名空间注册服务。在创建 AWS Cloud Map 私有 DNS 命名空间时，将自动创建 Route 53 私有托管区域。
- Amazon ECS 不支持向公有 DNS 命名空间注册服务。
- 为 服务发现 服务创建的 DNS 记录将始终注册任务的私有 IP 地址，而不是公有 IP 地址，即使使用公共命名空间也是如此。
- 服务发现 要求任务指定 awsvpc、bridge 或 host 网络模式（不支持 none）。
- 如果您的服务任务指定的任务定义使用 awsvpc 网络模式，您可以为每个服务任务创建 A 或 SRV 记录的任何组合。如果您使用 SRV 记录，则需要端口。
- 如果您的服务任务指定的任务定义使用 bridge 或 host 网络模式，则 SRV 记录是唯一受支持的 DNS 记录类型。为每个服务任务创建 SRV 记录。SRV 记录必须从任务定义中指定容器名称和容器端口组合。
- 服务发现 服务的 DNS 记录可以在 VPC 中查询。它们采用以下格式：<#### service name>.<#### namespace>。有关更多信息，请参阅 步骤 3：验证服务发现 (p. 313)。
- 在服务名称上执行 DNS 查询时，A 记录返回与您任务对应的一组 IP 地址。SRV 记录返回各个任务的一组 IP 地址和端口。
- 如果您有不超过 8 条正常的记录，Route 53 会向所有 DNS 查询提供所有正常记录。
- 如果所有记录都不正常，Route 53 会向 DNS 查询提供最多 8 条不正常的记录。
- 您可以为负载均衡器后面的 ECS 服务配置 服务发现，但 服务发现 流量始终路由至此任务而非负载均衡器。
- 服务发现 不支持使用 传统负载均衡器。
- 建议对服务发现服务使用由 Amazon ECS 管理的容器级别的运行状况检查。
  - HealthCheckCustomConfig — Amazon ECS 代表您管理运行状况检查。Amazon ECS 使用来自容器和运行状况检查的信息以及任务状态来更新 AWS Cloud Map 的运行状况。在创建 服务发现 服务时，使用 --health-check-custom-config 参数来指定。有关更多信息，请参阅 AWS Cloud Map API 参考中的 HealthCheckCustomConfig。

- 如果使用 Amazon ECS 控制台，则工作流程将为每个 ECS 服务创建一个服务发现服务。它将所有任务 IP 地址映射为 A 记录，或将任务 IP 地址和端口映射为 SRV 记录。
- 服务发现 只能在首次创建服务时进行配置。不支持更新现有服务以首次配置 服务发现 或更改当前配置。
- 必须手动清除使用服务发现时创建的 AWS Cloud Map 资源。有关更多信息，请参阅教程：使用 服务发现 创建服务 (p. 308)主题中的 步骤 4：清除 (p. 315)。

Amazon ECS 控制台体验

Amazon ECS 控制台中的服务创建和服务更新工作流程支持服务发现。

要创建使用服务发现的新 Amazon ECS 服务，请参阅创建服务 (p. 317)。

服务发现 定价

使用 Amazon ECS 服务发现的客户需要为 Route 53 资源和 AWS Cloud Map 发现 API 操作付费。这涉及到创建 Route 53 托管区域和查询服务注册表的成本。有关更多信息，请参阅 AWS Cloud Map 开发人员指南中的 AWS Cloud Map 定价。

Amazon ECS 执行容器级别的运行状况检查并将其公开到 AWS Cloud Map 自定义运行状况检查 API 操作。目前将此提供给客户没有任何额外成本。如果您为公开任务配置其他网络运行状况检查，则会向您收取这些运行状况检查的费用。

教程：使用 服务发现 创建服务

服务发现 已集成到 Amazon ECS 控制台的创建服务向导中。有关更多信息，请参阅创建服务 (p. 317)。

以下教程说明如何创建一个包含 Fargate 任务的 ECS 服务，该任务将 服务发现 与 AWS CLI 结合使用。

> **Note**
>
> 只有以下区域支持 Fargate 任务：

区域名称	区域
美国东部 (弗吉尼亚北部)	us-east-1
美国东部 (俄亥俄)	us-east-2
美国西部 (加利福尼亚北部)	us-west-1
美国西部 (俄勒冈)	us-west-2
亚太地区 (孟买)	ap-south-1
欧洲 (爱尔兰)	eu-west-1
欧洲 (伦敦)	eu-west-2
欧洲 (法兰克福)	eu-central-1
亚太地区 (东京)	ap-northeast-1
加拿大 (中部)	ca-central-1
亚太地区 (首尔)	ap-northeast-2
亚太地区 (新加坡)	ap-southeast-1
亚太地区 (悉尼)	ap-southeast-2

主题

- 先决条件 (p. 309)
- 步骤 1：创建服务发现资源 (p. 309)
- 步骤 2：创建 Amazon ECS 资源 (p. 310)
- 步骤 3：验证服务发现 (p. 313)
- 步骤 4：清除 (p. 315)

先决条件

本教程假设以下先决条件已完成：

- 安装并配置了最新版本的 AWS CLI。有关更多信息，请参阅安装 AWS 命令行接口。
- Amazon ECS 的设置 (p. 7) 中的步骤已完成。
- 您的 AWS 用户具有 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。有关更多信息，请参阅教程：为您的集群创建带有公有和私有子网的 VPC。

步骤 1：创建服务发现资源

使用以下步骤创建您的服务发现命名空间和服务发现服务。

创建服务发现资源

1. 在现有 VPC 之一中创建一个名为 `tutorial` 的私有服务发现命名空间：

```
aws servicediscovery create-private-dns-namespace --name tutorial --vpc vpc-abcd1234 --
region us-east-1
```

输出:

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. 使用来自之前输出的 `OperationId`，验证已成功创建私有命名空间。复制命名空间 ID，因为在后续命令中会使用它。

```
aws servicediscovery get-operation --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-
je6chs2e
```

输出:

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
            "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
```

```
}
```

3. 通过使用先前输出中的 NAMESPACE ID 来创建一个名为 myapplication 的服务发现服务。复制服务发现服务 ID，因为在后续命令中会使用它：

```
aws servicediscovery create-service --name myapplication --dns-config 'NamespaceId="ns-uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]' --health-check-custom-config
 FailureThreshold=1 --region us-east-1
```

输出:

```
{
    "Service": {
        "Id": "srv-utcrh6wavdkggqtk",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk",
        "Name": "myapplication",
        "DnsConfig": {
            "NamespaceId": "ns-uejictsjen2i4eeg",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 300
                }
            ]
        },
        "HealthCheckCustomConfig": {
            "FailureThreshold": 1
        },
        "CreatorRequestId": "e49a8797-b735-481b-a657-b74d1d6734eb"
    }
}
```

步骤 2：创建 Amazon ECS 资源

使用以下步骤创建您的 Amazon ECS 集群、任务定义和服务。

创建 Amazon ECS 资源

1. 创建要使用的名为 tutorial 的 Amazon ECS 集群：

```
aws ecs create-cluster --cluster-name tutorial --region us-east-1
```

输出:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/tutorial",
        "clusterName": "tutorial",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": []
    }
}
```

2. 注册与 Fargate 兼容的任务定义。它需要使用 awsvpc 网络模式。以下为用于本教程的示例任务定义。

首先，使用以下任务定义的内容创建名为 fargate-task.json 的文件：

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

然后，使用您创建的 fargate-task.json 文件注册任务定义：

```
aws ecs register-task-definition --cli-input-json file:///fargate-task.json --region us-
east-1
```

3. 使用您将要创建的 ECS 服务的内容创建名为 ecs-service-discovery.json 的文件。本示例使用在上一步中创建的任务定义。由于示例任务定义使用 awsvpc 网络模式，awsvpcConfiguration 是必需的。

```
{
    "cluster": "tutorial",
    "serviceName": "ecs-service-discovery",
    "taskDefinition": "tutorial-task-def",
    "serviceRegistries": [
       {
          "registryArn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk"
       }
    ],
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
```

```
            "securityGroups": [ "sg-abcd1234" ],
            "subnets": [ "subnet-abcd1234" ]
        }
    },
    "desiredCount": 1
}
```

创建 ECS 服务，指定 Fargate 启动类型和 LATEST 平台版本，它支持 服务发现：

```
aws ecs create-service --cli-input-json file://ecs-service-discovery.json --region us-east-1
```

输出:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:region:aws_account_id:service/ecs-service-discovery",
        "serviceName": "ecs-service-discovery",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [
            {
                "registryArn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-utcrh6wavdkggqtk"
            }
        ],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/tutorial-task-def:1",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/9223370516993140842",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/tutorial-task-def:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": 1519861634.965,
                "updatedAt": 1519861634.965,
                "launchType": "FARGATE",
                "platformVersion": "1.1.0",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-abcd1234"
                        ],
                        "securityGroups": [
                            "sg-abcd1234"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                }
```

```
            }
        ],
        "roleArn": "arn:aws:iam::aws_account_id:role/ECSServiceLinkedRole",
        "events": [],
        "createdAt": 1519861634.965,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-abcd1234"
                ],
                "securityGroups": [
                    "sg-abcd1234"
                ],
                "assignPublicIp": "ENABLED"
            }
        }
    }
}
```

步骤 3：验证服务发现

您可以通过查询您的服务发现信息来验证是否正确创建了所有内容。配置服务发现之后，可以使用 AWS Cloud Map API 操作或者在 VPC 中使用 `dig` 来查询，如下所述。

验证服务发现配置

1. 通过使用 服务发现 服务 ID，列出 服务发现 实例：

```
aws servicediscovery list-instances --service-id srv-utcrh6wavdkggqtk --region us-
east-1
```

输出:

```
{
    "Instances": [
        {
            "Id": "16becc26-8558-4af1-9fbd-f81be062a266",
            "Attributes": {
                "AWS_INSTANCE_IPV4": "172.31.87.2"
                "AWS_INSTANCE_PORT": "80",
                "AVAILABILITY_ZONE": "us-east-1a",
                "REGION": "us-east-1",
                "ECS_SERVICE_NAME": "ecs-service-discovery",
                "ECS_CLUSTER_NAME": "tutorial",
                "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
            }
        }
    ]
}
```

2. 通过服务发现命名空间和服务，使用其他参数来查询有关服务发现实例的详细信息：

```
aws servicediscovery discover-instances --namespace-name tutorial --service-
name myapplication --query-parameters ECS_CLUSTER_NAME=tutorial --region us-east-1
```

输出:

```
{
    "Instances": [
        {
            "InstanceId": "16becc26-8558-4af1-9fbd-f81be062a266",
            "NamespaceName": "tutorial",
            "ServiceName": "ecs-service-discovery",
            "HealthStatus": "HEALTHY",
            "Attributes": {
                "AWS_INSTANCE_IPV4": "172.31.87.2"
                "AWS_INSTANCE_PORT": "80",
                "AVAILABILITY_ZONE": "us-east-1a",
                "REGION": "us-east-1",
                "ECS_SERVICE_NAME": "ecs-service-discovery",
                "ECS_CLUSTER_NAME": "tutorial",
                "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
            }
        }
    ]
}
```

3. 在 Route 53 托管区域中为服务发现服务创建的 DNS 记录可以使用以下 AWS CLI 命令查询。

   使用命名空间 ID 获取有关命名空间的信息，其中包括 Route 53 托管区域 ID：

```
aws servicediscovery get-namespace --id ns-uejictsjen2i4eeg --region us-east-1
```

   输出:

```
{
    "Namespace": {
        "Id": "ns-uejictsjen2i4eeg",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:namespace/ns-
uejictsjen2i4eeg",
        "Name": "tutorial",
        "Type": "DNS_PRIVATE",
        "Properties": {
            "DnsProperties": {
                "HostedZoneId": "Z35JQ4ZFDRYPLV"
            }
        },
        "CreateDate": 1519777852.502,
        "CreatorRequestId": "9049a1d5-25e4-4115-8625-96dbda9a6093"
    }
}
```

4. 使用 Route 53 托管区域 ID 获取托管区域的资源记录集：

```
aws route53 list-resource-record-sets --hosted-zone-id Z35JQ4ZFDRYPLV --region us-
east-1
```

   输出:

```
{
    "ResourceRecordSets": [
        {
            "Name": "tutorial.",
            "Type": "NS",
            "TTL": 172800,
            "ResourceRecords": [
                {
```

```
                    "Value": "ns-1536.awsdns-00.co.uk."
                },
                {
                    "Value": "ns-0.awsdns-00.com."
                },
                {
                    "Value": "ns-1024.awsdns-00.org."
                },
                {
                    "Value": "ns-512.awsdns-00.net."
                }
            ]
        },
        {
            "Name": "tutorial.",
            "Type": "SOA",
            "TTL": 900,
            "ResourceRecords": [
                {
                    "Value": "ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com. 1
 7200 900 1209600 86400"
                }
            ]
        },
        {
            "Name": "myapplication.tutorial.",
            "Type": "A",
            "SetIdentifier": "16becc26-8558-4af1-9fbd-f81be062a266",
            "MultiValueAnswer": true,
            "TTL": 300,
            "ResourceRecords": [
                {
                    "Value": "172.31.87.2"
                }
            ]
        }
    ]
}
```

5. 您还可以使用 `dig`，从您 VPC 中的实例通过以下命令查询 DNS：

```
dig +short myapplication.tutorial
```

输出:

```
172.31.87.2
```

步骤 4：清除

完成本教程后，您应清除相关资源，以避免产生与未使用的资源相关的费用。

清除服务发现实例和 Amazon ECS 资源

1. 取消注册服务发现服务实例：

```
aws servicediscovery deregister-instance --service-id srv-utcrh6wavdkggqtk --instance-
id 16becc26-8558-4af1-9fbd-f81be062a266 --region us-east-1
```

输出:

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. 通过使用先前输出中的 OperationId，验证服务发现服务实例是否已成功取消注册：

```
aws servicediscovery get-operation --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-
jh0zimzv --region us-east-1
```

输出:

```
{
    "Operation": {
        "Id": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv",
        "Type": "DEREGISTER_INSTANCE",
        "Status": "SUCCESS",
        "CreateDate": 1525984073.707,
        "UpdateDate": 1525984076.426,
        "Targets": {
            "INSTANCE": "16becc26-8558-4af1-9fbd-f81be062a266",
            "ROUTE_53_CHANGE_ID": "C5NSRG1J4I1FH",
            "SERVICE": "srv-utcrh6wavdkggqtk"
        }
    }
}
```

3. 删除服务发现服务：

```
aws servicediscovery delete-service --id srv-utcrh6wavdkggqtk --region us-east-1
```

4. 删除服务发现命名空间：

```
aws servicediscovery delete-namespace --id ns-uejictsjen2i4eeg --region us-east-1
```

输出:

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. 通过使用先前输出中的 OperationId，验证服务发现命名空间是否已成功删除：

```
aws servicediscovery get-operation --operation-id c3ncqglftesw4ibgj5baz6ktaoh6cg4t-
jh0ztysj --region us-east-1
```

输出:

```
{
    "Operation": {
        "Id": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj",
        "Type": "DELETE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1525984602.211,
        "UpdateDate": 1525984602.558,
        "Targets": {
            "NAMESPACE": "ns-rymlehshst7hhukh",
            "ROUTE_53_CHANGE_ID": "CJP2A2M86XW3O"
        }
```

```
    }
}
```

6. 更新 Amazon ECS 服务，使预期数量为 0，从而允许您删除它：

```
aws ecs update-service --cluster tutorial --service ecs-service-discovery --desired-
count 0 --force-new-deployment --region us-east-1
```

7. 删除Amazon ECS服务：

```
aws ecs delete-service --cluster tutorial --service ecs-service-discovery --region us-
east-1
```

8. 删除 Amazon ECS 集群：

```
aws ecs delete-cluster --cluster tutorial --region us-east-1
```

创建服务

在创建 Amazon ECS 服务时，您将指定基本参数，用于定义构成服务的内容以及服务的行为方式。这些参数将创建一个服务定义。

您还可以选择配置其他功能，例如 Elastic Load Balancing 负载均衡器，用于在服务中的多个容器间分布流量。有关更多信息，请参阅服务负载均衡 (p. 284)。您必须验证容器实例是否能接收来自您的负载均衡器的流量。您可以允许流量从负载均衡器的安全组流到容器实例上的所有端口，以确保流量可到达使用动态分配的端口的任何容器。

以下文档将指导您完成 AWS 管理控制台中的创建服务向导的每个步骤。

主题

- 步骤 1：配置基本服务参数 (p. 317)
- 步骤 2：配置网络 (p. 319)
- 步骤 3：将您的服务配置为使用负载均衡器 (p. 319)
- 步骤 4：（可选）将服务配置为使用服务发现 (p. 323)
- 步骤 5：（可选）将服务配置为使用服务 Auto Scaling (p. 324)
- 步骤 6：查看并创建您的服务 (p. 326)

步骤 1：配置基本服务参数

所有服务都需要一些基本配置参数，用于定义服务（如要使用的任务定义、应运行服务的集群、应为服务放置的任务数等）。这称为服务定义。有关服务定义中定义的参数的更多信息，请参阅服务定义参数 (p. 272)。

该过程介绍如何创建包含必需的基本服务定义参数的服务。在配置了这些参数后，您可以创建您的服务，或继续进行可选的服务定义配置，如配置服务以使用负载均衡器。

配置基本服务定义参数

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏上，选择集群所在的区域。

3. 在导航窗格中，选择 Task Definitions，然后选择用于创建服务的任务定义。
4. 在 Task Definition name 页面上，选择用于创建服务的任务定义的修订。
5. 查看任务定义，然后依次选择操作和创建服务。
6. 在配置服务页面上，相应地填写以下参数：
   - Launch type (启动类型)：选择服务是应在 Fargate 基础设施上还是应在您维护的 Amazon EC2 容器实例上运行任务。有关更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。
   - Platform version (平台版本)：如果您选择了 Fargate 启动类型，请选择要使用的平台版本。
   - Cluster (集群)：选择要在其中创建服务的集群。
   - 服务名称：为您的服务键入唯一名称。
   - Service type (服务类型)：为您的服务选择一个计划策略。有关更多信息，请参阅服务计划程序概念 (p. 270)。
   - Number of tasks (任务数)：如果您选择了 `REPLICA` 服务类型，请键入要在您的集群中启动并维护的任务数。

     Note

     如果您的启动类型为 `EC2`，并且您的任务定义在容器实例上使用静态主机端口映射，则至少需要一个具有指定端口且在集群中可用于服务中的每个任务的容器实例。如果您的任务定义使用动态主机端口映射和 `bridge` 网络模式，则此限制不适用。有关详细信息，请参阅 portMappings (p. 55)。
   - 如果您使用的是 Rolling update (滚动更新) 部署类型，请填写以下参数：
     - 最小正常百分比：以服务的预期任务数的百分比形式指定部署期间必须保持在 `RUNNING` 状态的服务任务数的下限（向上取整到最近的整数）。例如，如果您的服务的预期任务数为 4，最小正常百分比为 50%，则计划程序可能在开始两个新任务之前停止两个现有任务以释放集群容量。服务任务的状态如果为 `RUNNING`，而且未使用负载均衡器，即为运行正常。服务任务的状态如果为 `RUNNING` 且确实使用了负载均衡器，同时托管任务的容器实例被负载均衡器报告为正常，即为运行正常。在控制台中，最小正常百分比的默认值为 50%，在 AWS CLI 或软件开发工具包中为 100%。
     - 最大百分比：以服务的预期任务数的百分比形式指定部署期间允许处于 `RUNNING` 或 `PENDING` 状态的服务任务数的上限（向下取整到最近的整数）。例如，如果您的服务的预期任务数为 4，最大百分比值为 200%，则计划程序可能在停止 4 个旧任务之前开始 4 个新任务。这样做的前提是具有执行此操作所需的集群资源。最大百分比的默认值为 200%。
7. 在 Deployments (部署) 页面上，相应地填写以下参数：
   - 对于 Deployment type (部署类型)，选择您的服务是应使用滚动更新部署还是应使用通过 AWS CodeDeploy 进行的蓝/绿部署。有关更多信息，请参阅Amazon ECS 部署类型 (p. 277)。
   - 如果您选择了蓝/绿部署类型，请为 Service role for CodeDeploy (CodeDeploy 的服务角色) 选择 AWS CodeDeploy 的 IAM 服务角色。有关更多信息，请参阅Amazon ECS CodeDeploy IAM 角色 (p. 411)。
8. （可选）如果您选择了 EC2 启动类型和 `REPLICA` 服务类型，则对于 Task Placement (任务放置)，可以使用任务放置策略和约束指定如何放置任务。从以下选项中进行选择：
   - AZ Balanced Spread (AZ 均衡分散) – 在各个可用区以及每个可用区中的各个容器实例中分配任务。
   - AZ Balanced BinPack (AZ 均衡装填) – 在各个可用区以及具有最低可用内存的容器实例中分配任务。
   - BinPack (装填) – 根据 CPU 或内存的最低可用量来分配任务。
   - One Task Per Host (每个主机一项任务) – 在每个容器实例中最多可放置服务的一个任务。
   - Custom (自定义) – 定义您自己的任务放置策略。有关示例，请参阅 Amazon ECS 任务放置 (p. 255)。

   有关更多信息，请参阅 Amazon ECS 任务放置 (p. 255)。
9. In the Task tagging configuration section, complete the following steps:

a. Select Enable ECS managed tags if you want Amazon ECS to automatically tag the tasks in the service with the Amazon ECS managed tags. For more information, see Tagging Your Amazon ECS Resources.

b. For Propagate tags from, select one of the following:

- Do not propagate – This option will not propagate any tags to the tasks in the service.
- Service – This option will propagate the tags specified on your service to each of the tasks in the service.
- Task Definitions – This option will propagate the tags specified in the task definition of a task to the tasks in the service.

> **Note**
>
> If you specify a tag with the same `key` in the Tags section, it will override the tag propagated from either the service or the task definition.

10. 在 Tags (标签) 部分中，为要与任务关联的每个标签指定键和值。有关更多信息，请参阅标记您的 Amazon ECS 资源。
11. 选择 Next step (下一步)，然后导航到步骤 2：配置网络 (p. 319)。

步骤 2：配置网络

如果服务的任务定义使用 `awsvpc` 网络模式，则您必须为您的服务配置 VPC、子网和安全组设置。

如果您的服务的任务定义没有使用 `awsvpc` 网络模式，则可以转到下一步步骤 3：将您的服务配置为使用负载均衡器 (p. 319)。

要为您的服务配置 VPC 和安全组设置，请执行以下操作：

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务配置过程操作。
2. 对于 Cluster VPC (集群 VPC)，如果选择 EC2 启动类型，请选择容器实例所在的 VPC。如果选择 Fargate 启动类型，请选择 Fargate 任务应使用的 VPC。确保您选择的 VPC 未配置为需要专用硬件租赁，因为 Fargate 任务对此不予支持。
3. 对于 Subnets，请为您的服务任务放置选择可用子网。

   > **Important**
   >
   > `awsvpc` 网络模式仅支持私有子网。由于任务没有收到公有 IP 地址，需要 NAT 网关来进行出站 Internet 访问。应通过负载均衡器来路由入站 Internet 流量。

4. 对于 Security groups，已经为您的服务任务创建了一个安全组，允许来自 Internet 的 HTTP 流量 (0.0.0.0/0)。要编辑此安全组的名称或规则，或选择现有安全组，请选择 Edit，然后修改安全组设置。
5. 对于自动分配公有 IP，选择是否让您的任务收到公有 IP 地址。如果您使用的是 Fargate 任务，则需要将公有 IP 地址分配给任务的弹性网络接口。网络接口必须具有到 Internet 的路由或可将请求路由到 Internet 的 NAT 网关，以便任务可以提取容器映像。
6. 如果您要将服务配置为使用负载均衡器，或者您使用的是绿/蓝部署类型，请继续步骤 3：将您的服务配置为使用负载均衡器 (p. 319)。如果您没有将服务配置为使用负载均衡器，则可以选择 None 作为负载均衡器类型，然后转到下一部分：步骤 5：（可选）将服务配置为使用服务 Auto Scaling (p. 324)。

步骤 3：将您的服务配置为使用负载均衡器

服务可配置为使用负载均衡器将传入流量分配到服务中的任务。如果您的服务使用滚动更新部署类型，这是可选的。如果您的服务使用蓝/绿部署类型，则需要使用应用程序负载均衡器或网络负载均衡器。

如果您没有将服务配置为使用负载均衡器，则可以选择 None 作为负载均衡器类型，然后转到下一部分：步骤 4：（可选）将服务配置为使用服务发现 (p. 323)。

如果已配置可用的 Elastic Load Balancing 负载均衡器，则可通过以下过程将其附加到服务，您也可以配置新的负载均衡器。有关更多信息，请参阅创建负载均衡器 (p. 289)。

> **Important**
>
> 在执行这些过程之前，您必须创建 Elastic Load Balancing 负载均衡器资源。

主题

- 为滚动更新部署类型配置负载均衡器 (p. 320)
- 为蓝/绿部署类型配置负载均衡器 (p. 322)

为滚动更新部署类型配置负载均衡器

如果您的服务的任务需要一段时间才能启动和响应 Elastic Load Balancing 运行状况检查，则您最长可以指定 2147483647 秒的运行状况检查宽限期。在此期间，服务计划程序忽略运行状况检查状态。此宽限期可防止服务计划程序将由于时间不足尚未启动的任务标记为不正常并停止它们。您的服务只有配置为使用负载均衡器时此设置才适用。

配置运行状况检查宽限期

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务配置过程操作。
2. 对于 Health check grace period (运行状况检查宽限期)：输入时间段（以秒为单位），在任务首次启动后，Amazon ECS 服务计划程序在此时间段内应忽略不正常的 Elastic Load Balancing 目标运行状况检查。

要将服务配置为使用负载均衡器，您必须选择要用于服务的负载均衡器类型。

选择负载均衡器类型

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务创建过程操作。
2. 对于负载均衡器类型，选择要用于您的服务的负载均衡器类型：

   应用程序负载均衡器

   允许容器使用动态主机端口映射，这使您能够在单个容器实例上使用同一端口放置多个任务。在单个负载均衡器上，多个服务可以通过基于规则的路由和路径使用同一个侦听器端口。

   网络负载均衡器

   允许容器使用动态主机端口映射，这使您能够在单个容器实例上使用同一端口放置多个任务。在单个负载均衡器上，多个服务可以通过基于规则的路由使用同一个侦听器端口。

   传统负载均衡器

   需要静态主机端口映射（每个容器实例只允许一个任务）；不支持基于规则的路由和路径。

   建议对 Amazon ECS 服务使用应用程序负载均衡器，以便利用这些服务可用的高级功能。
3. 对于 Select IAM role for service (为服务选择 IAM 角色)，选择 Create new role (创建新角色)以便为您的服务创建新角色，或选择要用于您的服务的现有 IAM 角色（默认情况下，此角色为 `ecsServiceRole`）。

> **Important**
>
> 如果选择使用现有 `ecsServiceRole` IAM 角色，则必须验证该角色是否拥有使用应用程序负载均衡器和 传统负载均衡器的适当权限。有关更多信息，请参阅Amazon ECS 服务计划程序 IAM 角色 (p. 410)。

4. 对于 ELB Name，选择要用于您的服务的负载均衡器的名称。此处只有与您之前选择的负载均衡器类型对应的负载均衡器可见。
5. 下一个步骤因服务的负载均衡器类型而异。如果选择了应用程序负载均衡器，请按照配置应用程序负载均衡器 (p. 321)中的步骤操作。如果选择了网络负载均衡器，请按照配置网络负载均衡器 (p. 321)中的步骤操作。如果选择了 传统负载均衡器，请按照配置传统负载均衡器 (p. 322)中的步骤操作。

配置应用程序负载均衡器

1. 对于用于负载均衡的容器，从任务定义中选择负载均衡器应将流量分发到的容器和端口组合，然后选择添加到负载均衡器。
2. 对于 Listener port (侦听器端口)，选择您在创建 应用程序负载均衡器 (p. 289)中创建的侦听器的侦听器端口和协议（如果适用）；或者选择 create new (新建) 以创建新侦听器，然后输入端口号并为 Listener protocol (侦听器协议) 选择端口协议。
3. 对于 Target group name，选择您在创建 应用程序负载均衡器 (p. 289)中创建的目标组（如果适用），或选择 create new 以创建新目标组。

   > **Important**
   >
   > 如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则目标组必须使用 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。

4. （可选）如果选择创建新目标组，请按照下面所示填写以下字段：
   - 对于 Target group name (目标组名)，为您提供了默认名称。
   - 对于 Target group protocol，输入用于将流量路由至您的任务的协议。
   - 对于 Path pattern，如果您的侦听器没有任何现有规则，则将使用默认路径模式 (`/`)。如果您的侦听器已经有默认规则，您必须输入与要发送到您的服务目标组的流量匹配的路径模式。例如，如果您的服务是名为 `web-app` 的 Web 应用程序，并且您希望与 `http://my-elb-url/web-app` 匹配的流量路由到您的服务，您应输入 `/web-app*` 作为您的路径模式。有关更多信息，请参阅Application Load Balancer 用户指南 中的 ListenerRules。
   - 对于 Health check path，输入负载均衡器应将运行状况检查 ping 发送到的路径。
5. 配置完应用程序负载均衡器之后，选择 Next step (下一步)。

配置网络负载均衡器

1. 对于用于负载均衡的容器，从任务定义中选择负载均衡器应将流量分发到的容器和端口组合，然后选择添加到负载均衡器。
2. 对于 Listener port (侦听器端口)，选择您在创建 应用程序负载均衡器 (p. 289)中创建的侦听器的侦听器端口和协议（如果适用）；或者选择 create new (新建) 以创建新侦听器，然后输入端口号并为 Listener protocol (侦听器协议) 选择端口协议。
3. 对于 Target group name，选择您在创建 应用程序负载均衡器 (p. 289)中创建的目标组（如果适用），或选择 create new 以创建新目标组。

   > **Important**
   >
   > 如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则目标组必须使用 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。

4. （可选）如果选择创建新目标组，请按照下面所示填写以下字段：

- 对于 Target group name (目标组名)，为您提供了默认名称。
- 对于 Target group protocol，输入用于将流量路由至您的任务的协议。
- 对于 Health check path，输入负载均衡器应将运行状况检查 ping 发送到的路径。

5. 配置完网络负载均衡器之后，选择 Next Step (下一步)。

配置传统负载均衡器

1. Health check port、Health check protocol 和 Health check path 字段是使用您在创建 传统负载均衡器 (p. 294)中配置的值（如果适用）预先填充的。您可在 Amazon EC2 控制台中更新这些设置。
2. 对于 Container for ELB health check，选择用于发送运行状况检查的容器。
3. 配置完传统负载均衡器之后，选择 Next step (下一步)。

为蓝/绿部署类型配置负载均衡器

要将使用蓝/绿部署类型的服务配置为使用负载均衡器，您必须使用应用程序负载均衡器或网络负载均衡器。

选择负载均衡器类型

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务创建过程操作。
2. 对于负载均衡器类型，选择要用于您的服务的负载均衡器类型：

   应用程序负载均衡器

   允许容器使用动态主机端口映射，这使您能够在单个容器实例上使用同一端口放置多个任务。在单个负载均衡器上，多个服务可以通过基于规则的路由和路径使用同一个侦听器端口。

   网络负载均衡器

   允许容器使用动态主机端口映射，这使您能够在单个容器实例上使用同一端口放置多个任务。在单个负载均衡器上，多个服务可以通过基于规则的路由使用同一个侦听器端口。

   建议对 Amazon ECS 服务使用应用程序负载均衡器，以便利用这些服务可用的高级功能。
3. 对于 Load balancer name (负载均衡器名称)，选择要用于您的服务的负载均衡器的名称。此处只有与您之前选择的负载均衡器类型对应的负载均衡器可见。
4. 下一个步骤因服务的负载均衡器类型而异。如果选择了应用程序负载均衡器，请按照配置应用程序负载均衡器 (p. 321)中的步骤操作。如果选择了网络负载均衡器，请按照配置网络负载均衡器 (p. 321)中的步骤操作。

为蓝/绿部署类型配置应用程序负载均衡器

1. 对于用于负载均衡的容器，从任务定义中选择负载均衡器应将流量分发到的容器和端口组合，然后选择添加到负载均衡器。
2. 对于 Production listener port (生产侦听器端口)，选择您在创建 应用程序负载均衡器 (p. 289)中创建的侦听器的侦听器端口和协议（如果适用）；或者选择 create new (新建) 以创建新侦听器，然后输入端口号并为 Production listener protocol (生产侦听器协议) 选择端口协议。
3. （可选）如果您希望在将流量路由到新任务集之前在负载均衡器上配置侦听器端口和协议来测试服务更新，请选择 Test listener (测试侦听器)。完成以下步骤：

   - 对于 Test listener port (测试侦听器端口)，选择要通过其测试流量的侦听器的侦听器端口和协议；或者选择 create new (新建) 以创建新测试侦听器，然后输入端口号并在 Test listener protocol (测试侦听器协议) 中选择端口协议。

4. 对于蓝/绿部署，需要两个目标组。每个目标组绑定到部署中的一个单独任务集。完成以下步骤：

   a. 对于 Target group 1 name (目标组 1 名称)，选择您在创建 应用程序负载均衡器 (p. 289)中创建的目标组（如果适用），或选择 create new (新建) 以创建新目标组。

      Important

      如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则目标组必须使用 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。

   b. （可选）如果选择创建新目标组，请按照下面所示填写以下字段：

      - 对于 Target group name，输入您的目标组的名称。
      - 对于 Target group protocol，输入用于将流量路由至您的任务的协议。
      - 对于 Path pattern，如果您的侦听器没有任何现有规则，则将使用默认路径模式 (/)。如果您的侦听器已经有默认规则，您必须输入与要发送到您的服务目标组的流量匹配的路径模式。例如，如果您的服务是名为 `web-app` 的 Web 应用程序，并且您希望与 `http://my-elb-url/web-app` 匹配的流量路由到您的服务，您应输入 `/web-app*` 作为您的路径模式。有关更多信息，请参阅Application Load Balancer 用户指南 中的 ListenerRules。
      - 对于 Health check path，输入负载均衡器应将运行状况检查 ping 发送到的路径。

   c. 为目标组 2 重复这些步骤。

   d. 配置完应用程序负载均衡器之后，选择 Next step (下一步)。导航到 步骤 4：（可选）将服务配置为使用服务发现 (p. 323)。

为蓝/绿部署类型配置网络负载均衡器

1. 对于用于负载均衡的容器，从任务定义中选择负载均衡器应将流量分发到的容器和端口组合，然后选择添加到负载均衡器。
2. 对于 Listener port (侦听器端口)，选择您在创建 应用程序负载均衡器 (p. 289)中创建的侦听器的侦听器端口和协议（如果适用）；或者选择 create new (新建) 以创建新侦听器，然后输入端口号并为 Listener protocol (侦听器协议) 选择端口协议。
3. 对于 Target group name，选择您在创建 应用程序负载均衡器 (p. 289)中创建的目标组（如果适用），或选择 create new 以创建新目标组。

   Important

   如果服务的任务定义使用 `awsvpc` 网络模式（为 Fargate 启动类型所需），则目标组必须使用 `ip` 而不是 `instance` 作为目标类型。这是因为使用 `awsvpc` 网络模式的任务与弹性网络接口而不是 Amazon EC2 实例关联。
4. （可选）如果选择创建新目标组，请按照下面所示填写以下字段：

   - 对于 Target group name，输入您的目标组的名称。
   - 对于 Target group protocol，输入用于将流量路由至您的任务的协议。
   - 对于 Health check path，输入负载均衡器应将运行状况检查 ping 发送到的路径。
5. 配置完网络负载均衡器之后，选择 Next Step (下一步)。导航到 步骤 4：（可选）将服务配置为使用服务发现 (p. 323)。

步骤 4：（可选）将服务配置为使用服务发现

Amazon ECS 服务可以选择启用 服务发现 集成，这样服务可以通过 DNS 搜索。有关更多信息，请参阅服务发现 (p. 306)。

如果您未将服务配置为使用服务发现，可以转到下一个部分步骤 5：（可选）将服务配置为使用服务 Auto Scaling (p. 324)。

配置 服务发现

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务配置过程操作。
2. 在 Configure network (配置网络) 页面上，选择 Enable 服务发现 integration (启用 服务发现 集成)。
3. 对于 Namespace (命名空间)，选择现有 Amazon Route 53 命名空间（如果已有），否则请选择 create new private namespace (创建新的私有命名空间)。
4. 如果创建新的命名空间，为 Namespace name (命名空间名称) 输入您命名空间的描述性名称。这是用于 Amazon Route 53 托管区域的名称。
5. 对于 Configure service (配置 服务发现 服务)，请选择创建新服务或者选择现有 服务发现 服务。
6. 如果创建新的 服务发现 服务，为 Service discovery name (服务发现名称) 输入 服务发现 服务的描述性名称。它用作要创建的 DNS 记录的前缀。
7. 如果希望为 服务发现 服务启用运行状况检查，请选择 Enable ECS task health propagation (启用 ECS 任务运行状况传播)。
8. 对于 DNS record type (DNS 记录类型)，选择要为服务创建的 DNS 记录类型。Amazon ECS 服务发现仅支持 A 和 SRV 记录，具体取决于任务定义指定的网络模式。有关这些记录类型的更多信息，请参阅 DnsRecord。
   - 如果您的服务任务指定的任务定义使用 `bridge` 或 `host` 网络模式，则只支持类型 SRV 记录。选择要与记录关联的容器名称和端口组合。
   - 如果您的服务任务指定的任务定义使用 `awsvpc` 网络模式，请选择 A 或 SRV 记录类型。如果选择了类型 A DNS 记录，请跳到下一步。如果选择了类型 SRV，请指定可以在其上找到该服务的端口或与该记录关联的容器名称和端口组合。
9. 对于 TTL，输入资源记录缓存存续时间 (TTL)，以秒计算。此值确定 DNS 解析程序和 Web 浏览器将记录集缓存多久。
10. 选择 Next step (下一步) 来继续导航到步骤 5：（可选）将服务配置为使用服务 Auto Scaling (p. 324)。

步骤 5：（可选）将服务配置为使用服务 Auto Scaling

您可以选择将 Amazon ECS 服务配置为使用 Auto Scaling 根据 CloudWatch 警报上调或下调其预期数量。

Amazon ECS 服务 Auto Scaling 支持以下类型的扩展策略：

- 目标跟踪扩展策略 (p. 300)— - 根据特定指标的目标值，增加或减少服务运行的任务数。这与恒温器保持家里温度的方式类似。您选择一个温度，恒温器将完成所有其他工作。
- 步进扩展策略 (p. 305) — 根据一组扩展调整，增加或减少服务运行的任务数，这些调整称为步进调整，将根据警报严重程度发生变化。

有关更多信息，请参阅 服务 Auto Scaling (p. 299)。

配置基本 服务 Auto Scaling 参数

1. 如果您尚未这样做，请按照步骤 1：配置基本服务参数 (p. 317)中的基本服务配置过程操作。
2. 在设置 Auto Scaling 页面上，选择配置服务 Auto Scaling 以调整您的服务的预期计数。
3. 对于 Minimum number of tasks (最小任务数)，输入供服务 Auto Scaling 使用的任务数的下限。您的服务的预期数量不会自动调整到低于此数量。
4. 对于 Desired number of tasks (预期任务数)，此字段是使用您之前输入的值预先填充的。此时您可更改服务的预期数量，但此值必须介于此页面上指定的最小任务数和最大任务数之间。
5. 对于 Maximum number of tasks (最大任务数)，输入供服务 Auto Scaling 使用的任务数的上限。您的服务的预期数量不会自动调整到高于此数量。
6. 对于 IAM role for Service Auto Scaling (与服务 Auto Scaling 配合使用的 IAM 角色)，选择一个 IAM 角色来授权 Application Auto Scaling 服务代表您调整服务的预期数量。如果您之前未创建此类角

色，请选择 Create new role，系统将为您创建此角色。为了方便将来参考，为您创建的角色名为 `ecsAutoscaleRole`。有关更多信息，请参阅 Amazon ECS 服务 Auto Scaling IAM 角色 (p. 414)。

7. 以下过程提供了用于为您的服务创建目标跟踪或分步扩展策略的步骤。选择您需要的扩展策略类型。

以下步骤将帮助您创建目标跟踪扩展策略和可用于为您的服务触发扩展活动的 CloudWatch 警报。您可以创建扩展警报来增加服务的预期数量，并创建缩减警报来减少服务的预期数量。

为服务配置目标跟踪扩展策略

1. 对于 Scaling policy type (扩展策略类型) 选择 Target tracking (目标跟踪)。
2. 对于 Policy name (策略名称)，请键入策略的描述性名称。
3. 对于 ECS service metric (ECS 服务指标)，请选择要跟踪的指标。
4. 对于 Target value (目标值)，请输入策略应保持的指标值。
5. 对于 Scale-out cooldown period (扩展冷却时间)，请输入在一个扩展活动完成后、另一个扩展活动开始之前的时间量 (秒)。在此期间，已启动的资源不影响 Auto Scaling 组指标。
6. 对于 Scale-in cooldown period (缩减冷却时间)，请输入在一个缩减活动完成之后、另一个缩减活动开始之前的时间量（秒）。在此期间，已启动的资源不影响 Auto Scaling 组指标。
7. （可选）要为此策略禁用缩减操作，请选择 Disable scale-in (禁用缩减)。这样，您稍后就可以为缩减创建单独的扩展策略。
8. 选择下一步。

以下步骤将帮助您创建分步扩展策略和可用于为您的服务触发扩展活动的 CloudWatch 警报。您可以创建 Scale out 警报来增加服务的预期数量，并创建 Scale in 警报来减少服务的预期数量。

为您的服务配置分步扩展策略

1. 对于 Scaling policy type (扩展策略类型) 选择 Step scaling (分步扩展)。
2. 对于 Policy name (策略名称)，请键入策略的描述性名称。
3. 对于 Execute policy when (执行策略的时间)，选择要用于扩展或缩减您的服务的 CloudWatch 警报。

   您可使用之前创建的现有 CloudWatch 警报，也可选择创建新警报。Create new alarm (创建新警报) 工作流程可让您创建基于要创建的服务的 `CPUUtilization` 和 `MemoryUtilization` 的 CloudWatch 警报。要使用其他指标，您可在 CloudWatch 控制台中创建一个警报，然后返回本向导以选择该警报。
4. （可选）如果您选择创建新警报，请完成以下步骤。
   a. 对于 Alarm name，为您的警报输入一个描述性名称。例如，如果您的警报应在服务 CPU 使用率超过 75% 时触发，则可调用警报 *`service_name`*`-cpu-gt-75`。
   b. 对于 ECS service metric，选择要用于警报的服务指标。有关更多信息，请参阅 服务利用率 (p. 347)。
   c. 对于 Alarm threshold，输入以下信息以配置您的警报：
      - 为警报选择 CloudWatch 统计数据（默认值 Average (平均值) 适用于很多情况）。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的统计数据 。
      - 为您的警报选择比较运算符并输入比较运算符检查所针对的值（例如，> 和 `75`）。
      - 输入触发警报前的连续时间段的数量和总长度。例如，2 个连续的 5 分钟时间段需要 10 分钟才会触发警报。由于您的 Amazon ECS 任务可快速扩展和缩减，所以应考虑使用少量的连续时间段和较短的时间段长度以尽快响应警报。
   d. 选择 Save。
5. 对于 Scaling action，输入以下信息以配置您的服务对警报的响应方式：
   - 选择为您的服务增加预期数量、减少预期数量还是设置具体预期数量。

- 如果您选择增加或减少任务，请输入触发扩展操作后要增加或减少的任务数（或现有任务的百分比）。如果您选择设置预期数量，请输入在触发扩展操作时应为您的服务设置的预期数量。
- （可选）如果您选择增加或减少任务，请选择将之前的值用作整数还是现有预期数量的百分比值。
- 输入您的步进扩展调整的下限。默认情况下，对于您的第一次扩展操作，此值是触发您的警报的指标数量。例如，以下扩展操作将在 CPU 使用率高于 75% 时增加 100% 的现有预期数量。

6. (可选) 您可重复 Step 5 (p. 325) 以便为单个警报配置多个扩展操作 (例如，在 CPU 使用率介于 75 和 85% 之间时增加 1 个任务，在 CPU 使用率超过 85% 时增加 2 个任务)。
7. （可选）如果您选择增大或减小现有预期数量的百分比，请在 Add tasks in increments of *N* task(s) 中输入最小增量值。
8. 对于 Cooldown period，输入两次扩展操作之间的秒数。
9. 对 Scale in (缩减) 策略重复Step 1 (p. 325) 至Step 8 (p. 326)，然后选择 Save (保存)。
10. 选择 Next step (下一步) 来继续导航到步骤 6：查看并创建您的服务 (p. 326)。

步骤 6：查看并创建您的服务

在配置基本服务定义参数并（可选）配置服务的联网、负载均衡器、服务发现和自动扩展后，您可查看您的配置。然后，选择 Create Service (创建服务) 完成服务创建。

> **Note**
>
> 在创建服务后，在服务定义中指定的目标组 ARN 或负载均衡器名称、容器名称和容器端口是不可变的。您无法添加、删除或更改现有服务的负载均衡器配置。如果您更新服务的任务定义，则创建服务时指定的容器名称和容器端口必须保留在任务定义中。

更新服务

您可以更新正在运行的服务以更改该服务维护的任务数、任务使用哪个任务定义，如果您的任务使用的是 Fargate 启动类型，也可以更改服务使用的平台版本。如果您有需要更多容量的应用程序，则可以扩展服务。如果您有要缩减的未使用容量，则可以减少服务中的所需任务的数量并释放资源。

如果您已更新应用程序的 Docker 映像，则可使用该映像创建新的任务定义并将此定义部署到您的服务。

> **Note**
>
> 如果您使用与服务的现有任务定义中的相同标记更新了 Docker 映像 (例如，`my_image:latest`)，则无需创建任务定义的新版本。您可以使用以下过程更新服务，保留服务的当前设置，以及选择 Force new deployment (强制实施新部署)。由部署启动的新任务将在启动时，从存储库中提取当前映像/标记组合。在指定 `LATEST` 来更新 Fargate 以使用最新的平台版本时，也会使用 Force new deployment (强制实施新部署) 选项。例如，如果您指定了 `LATEST` 并且您正在运行的任务使用 `1.0.0` 平台版本，而您希望使用较新的平台版本重新启动时。

服务计划程序将使用最小正常百分比和最大百分比参数（在服务的部署配置中）确定部署策略。

如果服务使用滚动更新 (`ECS`) 部署类型，最小正常百分比以预期任务数的百分比形式表示部署期间必须保持在 `RUNNING` 状态的服务中的任务数的下限（向上取整到最近的整数）。如果服务包含使用 EC2 启动类型的任务，则在任何容器实例处于 `DRAINING` 状态时，此参数也适用。此参数使您不必使用额外的集群容量就能部署。例如，如果您的服务的预期任务数为 4，最小正常百分比为 50%，则计划程序可能在开始两个新任务

之前停止两个现有任务以释放集群容量。服务任务的状态如果为 RUNNING，而且未使用负载均衡器，即为运行正常。如果使用负载均衡器的服务任务处于 RUNNING 状态，并且该负载均衡器将其报告为正常，则这些服务任务被视为正常。最小正常百分比的默认值为 100%。

如果服务使用滚动更新 (ECS) 部署类型，最大百分比参数以预期任务数的百分比形式表示部署期间允许处于 RUNNING 或 PENDING 状态的服务中的任务数的上限（向下取整到最近的整数）。如果服务包含使用 EC2 启动类型的任务，则在任何容器实例处于 DRAINING 状态时，此参数也适用。此参数使您能够定义部署批次大小。例如，如果您的服务的预期任务数为 4，最大百分比值为 200%，则计划程序可能在停止 4 个旧任务之前开始 4 个新任务。这样做的前提是具有执行此操作所需的集群资源。最大百分比的默认值为 200%。

如果服务使用蓝/绿 (CODE_DEPLOY) 部署类型和使用 EC2 启动类型的任务，最小正常百分比和最大百分比值将设置为默认值。它们只用于定义在容器实例处于 DRAINING 状态时保持在 RUNNING 状态的服务中的任务数的下限和上限。如果服务中的任务使用 Fargate 启动类型，则不使用最小正常百分比和最大百分比值。目前，它们在描述您的服务时可见。

当服务计划程序在更新期间替换某个任务时，服务首先会从负载均衡器（如果使用了）中删除此任务并等待连接耗尽。然后，将向此任务中运行的容器发出 docker stop 的等效项。这将产生 SIGTERM 信号和 30 秒的超时，此后，将发送 SIGKILL 并强制停止容器。如果容器正常处理了 SIGTERM 信号并在收到信号后的 30 秒内退出，则不会发送任何 SIGKILL 信号。服务计划程序将启动并停止您的最小正常百分比和最大百分比设置定义的任务。

> **Important**
>
> 如果要在任务定义中更改容器使用的端口，您可能需要更新容器实例安全组以使用更新后的端口。如果您的服务使用一个负载均衡器，则无法更改创建该负载均衡器时为服务定义的负载均衡器配置。如果您更新服务的任务定义，则创建服务时指定的容器名称和容器端口必须保留在任务定义中。
> 要更改与服务负载均衡器配置关联的负载均衡器名称、容器名称或容器端口，您必须创建新服务。Amazon ECS 不会自动更新与 Elastic Load Balancing 负载均衡器或 Amazon ECS 容器实例关联的安全组。

更新运行服务

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏上，选择集群所在的区域。
3. 在导航窗格中，选择 Clusters。
4. 在集群页面上，选择您的服务驻留的集群的名称。
5. 在 Cluster: ***name*** 页面上，选择 Services。
6. 选中要更新的服务左侧的框并选择 Update。
7. 在配置服务页面上，您的服务信息已预先填充。更改任务定义、平台版本、部署配置或所需任务的数量 (或这些设置的任意组合)，然后选择下一步。

   > **Note**
   >
   > 要让服务使用新更新的 Docker 映像以及与现有任务定义中相同的标签（例如，my_image:latest），或者保留服务的当前设置，请选择 Force new deployment (强制实施新部署)。由部署启动的新任务将在启动时，从存储库中提取当前映像/标记组合。在指定 LATEST 来更新 Fargate 以使用最新的平台版本时，也会使用 Force new deployment (强制实施新部署) 选项。例如，如果您指定了 LATEST 并且您正在运行的任务使用 1.0.0 平台版本，而您希望使用较新的平台版本重新启动时。
8. 在 Configure deployments (配置部署) 页面上，如果您的服务使用蓝/绿部署类型，则会预先填充您的服务部署的组件。确认以下设置。

   a. 对于 Application name (应用程序名称)，选择您的服务属于的 CodeDeploy 应用程序。
   b. 对于 Deployment group name (部署组名称)，选择您的服务属于的 CodeDeploy 部署组。
   c. 选择要作为服务部署的新修订的一部分执行的部署生命周期事件挂钩及关联的 Lambda 函数。可用的生命周期挂钩有：

- BeforeInstall – 在创建替换任务集之前，使用此部署生命周期事件挂钩来调用 Lambda 函数。在此生命周期事件时运行的 Lambda 函数结果不会触发回滚。
- AfterInstall – 在创建替换任务集之后，使用此部署生命周期事件挂钩来调用 Lambda 函数。在此生命周期事件时运行的 Lambda 函数结果会触发回滚。
- BeforeAllowTraffic – 在生产流量重新路由到替换任务集之前，使用此部署生命周期事件挂钩来调用 Lambda 函数。在此生命周期事件时运行的 Lambda 函数结果会触发回滚。
- AfterAllowTraffic – 在生产流量重新路由到替换任务集之后，使用此部署生命周期事件挂钩来调用 Lambda 函数。在此生命周期事件时运行的 Lambda 函数结果会触发回滚。

有关生命周期挂钩的更多信息，请参阅 AWS CodeDeploy User Guide 中的 AppSpec“hooks”部分。

9. 选择下一步。
10. 在配置网络页面上，您的网络信息已预先填充。在 Load balancing (负载均衡) 部分中，如果您的服务使用的是蓝/绿部署类型，请选择要与目标组关联的侦听器。更改运行状况检查宽限期 (如果需要)，然后选择下一步。
11. （可选）您可使用服务 Auto Scaling 自动扩展和缩减服务以响应 CloudWatch 警报。
    a. 在 Optional configurations (可选配置) 下，选择 Configure 服务 Auto Scaling (配置服务 Auto Scaling)。
    b. 继续执行步骤 5：（可选）将服务配置为使用服务 Auto Scaling (p. 324)。
    c. 完成该节中的步骤，然后返回。
12. 选择 Update Service 以完成并更新服务。

删除服务

您可以使用控制台删除 Amazon ECS 服务。在删除之前，服务自动缩减到零。如果您有与服务关联的负载均衡器或 服务发现 资源，则它们不受服务删除的影响。要删除 Elastic Load Balancing 资源，请参阅下列主题之一，具体取决于负载均衡器类型：删除应用程序负载均衡器或删除网络负载均衡器。要删除您的 服务发现 资源，请按以下过程操作。

删除 Amazon ECS 服务

使用以下过程删除 Amazon ECS 服务。

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航栏上，选择集群所在的区域。
3. 在导航窗格中，选择 Clusters (集群)，然后选择您的服务所在的集群的名称。
4. 在 Cluster: ***name*** 页面上，选择 Services。
5. 选中要更新的服务左侧的框并选择 Delete。
6. 输入文本短语来确认服务删除，然后选择删除。

删除 服务发现 资源 (AWS CLI)

要删除剩余的服务发现资源，您可以使用 AWS CLI 删除服务发现服务和服务发现命名空间。

1. 确保安装并配置了最新版本的 AWS CLI。有关安装或升级 AWS CLI 的更多信息，请参阅安装 AWS 命令行界面。
2. 检索要删除的 服务发现 服务的 ID。

```
aws servicediscovery list-services --region <region_name>
```

> **Note**
>
> 如果未返回 服务发现 服务，请继续执行步骤 4。

3. 通过使用先前输出中的 服务发现 服务 ID 来删除服务。

```
aws servicediscovery delete-service --id <service_discovery_service_id> --
region <region_name>
```

4. 检索要删除的 服务发现 命名空间的 ID。

```
aws servicediscovery list-namespaces --region <region_name>
```

5. 通过使用先前输出中的 服务发现 命名空间 ID 来删除命名空间。

```
aws servicediscovery delete-namespace --id <service_discovery_namespace_id> --
region <region_name>
```

服务限制逻辑

Amazon ECS 服务计划程序包含限制服务任务在反复启动失败后再启动的频率逻辑。

如果一个 ECS 服务的任务总是无法进入 `RUNNING` 状态（直接从 `PENDING` 跳到 `STOPPED`），则后续重启尝试间隔的时间会逐渐拉长，最多达到 15 分钟。最长时间将来可能有所变化，并非一成不变。这种机制减少了无法重启的任务对 Amazon ECS 集群资源或 Fargate 基础设施成本的不利影响。如果您的服务触发了限制逻辑，则您会收到以下服务事件消息 (p. 584)：

```
(service service-name) is unable to consistently start tasks successfully.
```

Amazon ECS 永远不会阻止失败的服务重试，也不会尝试修改服务，除了延长重启的间隔时间外。服务限制逻辑不提供任何用户可调参数。

如果您将服务更新为使用新的任务定义，则您的服务会立即返回到正常的无限制状态。有关更多信息，请参阅 更新服务 (p. 326)。

以下是触发此逻辑的一些常见原因：

- 集群中缺乏用来托管任务的资源，如端口、内存或 CPU 单位。在这种情况下，您还会看到资源不足服务事件消息 (p. 582)。
- Amazon ECS 容器代理无法提取您的任务 Docker 映像。这可能是因为容器映像名称、映像、标签错误，或者缺少私有注册表身份验证或权限。在这种情况下，您还会在停止的任务错误 (p. 579)中看到 `CannotPullContainerError`。
- 您的容器实例上的磁盘空间不足，无法创建容器。在这种情况下，您还会在停止的任务错误 (p. 579)中看到 `CannotCreateContainerError`。有关更多信息，请参阅`CannotCreateContainerError: API error (500): devmapper` (p. 586)。

> **Important**
>
> 进入 `RUNNING` 状态后停止的任务不会触发限制逻辑或相关服务事件消息。例如，如果因某个服务的 Elastic Load Balancing 运行状况检查失败而导致某个任务被标记为运行状况不佳，并且 Amazon ECS 取消注册了该服务并终止了该任务，则不会触发限制。即使任务的容器命令立即退出并伴随非零退出代码出现，该任务也已转为 `RUNNING` 状态。由于命令错误而立刻失败的任务不会触发限制或服务事件消息。

资源和标签

为 Amazon ECS 资源（包括任务定义、集群、任务、服务和容器实例）分配了 Amazon 资源名称 (ARN) 和唯一的资源标识符 (ID)。可以用您定义的值标记这些资源，来帮助您组织和识别它们。

以下主题介绍了资源和标签，以及如何使用它们。

目录

- 标记您的 Amazon ECS 资源 (p. 330)
- Amazon ECS 使用率报告 (p. 335)

标记您的 Amazon ECS 资源

为了方便管理您的 Amazon ECS 任务、服务、任务定义、集群和容器实例，您可以选择通过标签 的形式向每个资源分配您自己的元数据。本主题介绍标签并说明如何创建标签。

> Important
>
> 要使用此功能，需要您选择使用新 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。

内容

- 有关标签的基本知识 (p. 330)
- 标记您的资源 (p. 331)
- 标签限制 (p. 331)
- 标记资源以便于计费 (p. 332)
- 通过控制台使用标签 (p. 332)
- 通过 CLI 或 API 使用标签 (p. 334)

有关标签的基本知识

标签是为 AWS 资源分配的标记。每个标签都包含您定义的一个键 和一个可选值。

标签可让您按各种标准（例如用途、所有者或环境）对 AWS 资源进行分类。这在您具有相同类型的许多资源时会很有用 — 您可以根据分配给资源的标签快速识别特定资源。例如，您可以为账户的 Amazon ECS 容器实例定义一组标签，以帮助您跟踪每个容器实例的所有者和堆栈级别。

我们建议您针对每类资源设计一组标签，以满足您的需要。使用一组连续的标签键，管理资源时会更加轻松。您可以根据添加的标签搜索和筛选资源。

标签对 Amazon ECS 没有任何语义意义，应严格按字符串进行解析。同时，标签不会自动分配至您的资源。您可以修改标签的密钥和值，还可以随时删除资源的标签。您可以将标签的值设为空的字符串，但是不能将其设为空值。如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。如果删除资源，资源的所有标签也会被删除。

可以使用 AWS 管理控制台、AWS CLI 和 Amazon ECS API 处理标签。

如果您使用的是 AWS Identity and Access Management (IAM)，则可以控制 AWS 账户中的哪些用户拥有创建、编辑或删除标签的权限。

标记您的资源

您可以标记新的或现有的 Amazon ECS 任务、服务、任务定义和集群。

如果您使用的是 Amazon ECS 控制台，则可以在创建新资源时对其应用标签，或随时在相关资源页面上使用 Tags (标签) 选项卡对现有资源应用标签。Propagate tags from (传播以下来源的标签) 选项可在运行任务以将标签从任务定义复制到任务或在创建服务以将标签从服务或任务定义复制到服务中的任务时使用。

如果您使用的是 Amazon ECS API、AWS CLI 或 AWS 开发工具包，则可以使用相关 API 操作上的 `tags` 参数对新资源应用标签，或使用 `TagResource` API 操作对现有资源应用标签。有关更多信息，请参阅 TagResource。propagateTags 参数可在运行任务以将标签从任务定义复制到任务或在创建服务以将标签从服务或任务定义复制到服务中的任务时使用。有关更多信息，请参阅 RunTask 和 CreateService。

此外，某些资源创建操作允许您在创建资源时为其指定标签。如果无法在资源创建期间应用标签，系统会回滚资源创建过程。这样可确保要么创建带有标签的资源，要么根本不创建资源，即任何时候都不会创建出未标记的资源。通过在创建时标记资源，您不需要在资源创建后运行自定义标记脚本。

下表描述了可以标记的 Amazon ECS 资源以及可在创建时标记的资源。

Amazon ECS 资源标记支持

资源	支持标签	支持标签传播	支持在创建时标记（Amazon ECS API、AWS CLI、AWS 开发工具包）
Amazon ECS 任务	是	是，从任务定义。	是
Amazon ECS 服务	是	是，从任务定义或服务到服务中的任务。	是
Amazon ECS 任务定义	是	否	是
Amazon ECS 集群	是	否	是
Amazon ECS 容器实例	是	是，从 Amazon EC2 实例。有关更多信息，请参阅 向容器实例添加标签 (p. 333)。	是

标签限制

下面是适用于标签的基本限制：

- 每个资源的最大标签数 – 50
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大键长度 – 128 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 – 256 个 Unicode 字符（采用 UTF-8 格式）
- 如果您的标记方案针对多个服务和资源使用，请记得其他服务可能对允许使用的字符有限制。通常允许使用的字符包括：可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：+ - = . _ : / @。
- 标签键和值区分大小写。
- 请不要使用 `aws:`、`AWS:` 或任何大写或小写组合（例如，键或值的前缀），因为它将保留以供 AWS 使用。您无法编辑或删除带此前缀的标签键或值。具有此前缀的标签不计入每个资源的标签数限制。

标记资源以便于计费

在启用 Amazon ECS 托管标签后，Amazon ECS 将自动使用集群名称标记所有新启动的任务。对于属于服务的任务，还将使用服务名称标记它们。在成本和使用率报告中启用这些托管标签后，在查看成本分配时这些托管标签非常有帮助。有关更多信息，请参阅Amazon ECS 使用率报告 (p. 335)。

如需查看组合资源的成本，请按具有相同标签键值的资源组织您的账单信息。例如，您可以将特定的应用程序名称用作几个资源的标签，然后组织账单信息，以查看在数个服务中的使用该应用程序的总成本。有关使用标签设置成本分配报告的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的月度成本分配报告。

> Important
>
> 要使用此功能，需要您选择使用新 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。
>
> Note
>
> 如果您已启用报告，则可以在 24 小时后查看当月的数据。

通过控制台使用标签

使用 Amazon ECS 控制台，您可以管理与新的或现有的任务、服务、任务定义、集群或容器实例相关的标签。

当您在 Amazon ECS 控制台中选择特定资源页面时，它会显示这些资源列表。例如，如果您从导航窗格中选择 Clusters (集群)，则控制台会显示 Amazon ECS 集群列表。当您从其中一个列表中选择一种资源时（例如，特定集群），如果该资源支持标签，则您可以在 Tags (标签) 选项卡上查看和管理其标签。

内容

- 在启动过程中为单个资源添加标签 (p. 332)
- 为单个资源添加和删除标签 (p. 332)
- 向容器实例添加标签 (p. 333)

在启动过程中为单个资源添加标签

以下资源允许您在创建该资源时指定标签。

任务	控制台
运行一个或多个任务。	正在运行的任务 (p. 251)
创建服务。	创建服务 (p. 317)
注册任务定义。	创建任务定义 (p. 45)
创建集群。	创建集群 (p. 38)
运行一个或多个容器实例。	启动 Amazon ECS 容器实例 (p. 165)

为单个资源添加和删除标签

Amazon ECS 允许您直接从资源的页面中添加或删除与集群、服务、任务和任务定义相关的标签。有关标记容器实例的信息，请参阅向容器实例添加标签 (p. 333)。

向单个资源添加标签

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择资源类型（例如，Clusters (集群)）。
4. 从资源列表中选择资源，然后依次选择 Tags (标签) 和 Edit (编辑)。
5. 在 Edit Tags (编辑标签) 对话框中，为每个标签指定键和值，然后选择 Save (保存)。

删除单个资源的标签

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择资源类型（例如，Clusters (集群)）。
4. 从资源列表中选择资源，然后依次选择 Tags (标签) 和 Edit (编辑)。
5. 在 Edit Tags (编辑标签) 页面上，选择要删除的每个标签对应的 Delete (删除) 图标，然后选择 Save (保存)。

向容器实例添加标签

您可以使用以下方法之一将标签与您的容器实例相关联：

- 方法 1 – 在使用 Amazon EC2 API、CLI 或控制台创建容器实例时，通过使用容器代理配置参数 `ECS_CONTAINER_INSTANCE_TAGS` 将用户数据传递到实例来指定标签。这只创建与 Amazon ECS 中的容器实例关联的标签，而无法使用 Amazon EC2 API 列出这些标签。有关更多信息，请参阅 使用 Amazon EC2 用户数据引导启动容器实例 (p. 168)。

  > **Important**
  >
  > 如果您使用 Amazon EC2 Auto Scaling 组启动容器实例，则应使用 ECS_CONTAINER_INSTANCE_TAGS 代理配置参数添加标签。这是将标签添加到使用 Auto Scaling 组所启动的 Amazon EC2 实例的方式决定的。

  下面是将标签与您的容器实例相关联的用户数据脚本示例：

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  EOF
  ```

- 方法 2 – 当使用 Amazon EC2 API、CLI 或控制台在创建容器实例时，使用 `TagSpecification.N` 参数指定标签，然后使用容器代理配置参数 `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` 将用户数据传递到实例，这会将标签从 Amazon EC2 传播到 Amazon ECS

  下面是一个用户数据脚本示例，该脚本将传播与 Amazon EC2 实例关联的标签，以及向名为 `MyCluster` 的集群注册实例：

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

  要提供访问权限以允许容器实例标签从 Amazon EC2 传播到 Amazon ECS，请手动将以下权限作为内链策略添加到 Amazon ECS 容器实例 IAM 角色。有关更多信息，请参阅添加和删除 IAM 策略。

- `ec2:DescribeTags`

添加权限的示例内联策略如下所示

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
             "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
    ]
}
```

通过 CLI 或 API 使用标签

使用以下命令添加、更新、列出和删除资源标签。相应文档提供了示例。

Amazon ECS 资源标记支持

任务	AWS CLI	API 操作
添加或覆盖一个或多个标签。	tag-resource	TagResource
删除一个或多个标签。	untag-resource	UntagResource

以下示例说明如何使用 AWS CLI 标记或取消标记资源。

示例 1：标记现有集群

以下命令标记现有集群。

```
aws ecs tag-resource --resource-arn resource_ARN --tags key=stack,value=dev
```

示例 2：取消标记现有集群

以下命令从现有集群删除标签。

```
aws ecs untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

示例 3：列出资源的标签

以下命令列出与现有资源关联的标签。

```
aws ecs list-tags-for-resource --resource-arn resource_ARN
```

某些资源创建操作允许您在创建资源时指定标签。以下操作支持在创建时进行标记。

任务	AWS CLI	适用于 Windows PowerShell 的 AWS 工具	API 操作
运行一个或多个任务。	run-task	Start-ECSTask	RunTask

任务	AWS CLI	适用于 Windows PowerShell 的 AWS 工具	API 操作
创建服务。	create-service	New-ECSService	CreateService
注册任务定义。	register-task-definition	Register-ECSTaskDefinition	RegisterTaskDefinition
创建集群。	create-cluster	New-ECSCluster	CreateCluster
运行一个或多个容器实例。	run-instances	New-EC2Instance	RunInstances

以下示例说明如何在创建资源时应用标签。

示例 1：创建集群并应用标签

以下命令创建一个名为 `devcluster` 的集群并添加键为 `team` 且值为 `devs` 的标签。

```
aws ecs create-cluster --cluster-name devcluster --tags key=team,value=devs
```

示例 2：创建服务并应用标签

以下命令创建一个名为 `application` 的服务并添加键为 `stack` 且值为 `dev` 的标签。

```
aws ecs create-service --service-name application --task-definition task-def-app --tags
 key=stack,value=dev
```

示例 3：创建带标签的服务并将标签传播到服务中的任务。

`--propagateTags` 参数可用于将标签从任务定义或服务复制到服务中的任务。以下命令创建具有标签的服务并将标签传播到该服务中的任务。

```
aws ecs create-service --service-name application --task-definition task-def-app --tags
 key=stack,value=dev --propagateTags Service
```

Amazon ECS 使用率报告

AWS 提供了称为 Cost Explorer 的免费报告工具，该工具可让您分析 Amazon ECS 资源的成本和使用率。

Cost Explorer 是一款免费工具，可用于查看使用率和成本的图表。您可以查看过去 13 个月的数据，并预测您在接下来三个月内可能产生的费用。您可以使用 Cost Explorer 查看有关您一段时间内在 AWS 资源方面的费用的模式、确定需要进一步查询的方面以及查看可用于了解您的成本的趋势。您还可以指定数据的时间范围，并按天或按月查看时间数据。

成本和使用率报告中的计量数据显示跨所有 Amazon ECS 任务的使用率。计量数据包括运行的每个任务的 `vCPU-Hours` 和内存 `GB-Hours`。提供数据的方式取决于任务的启动类型，如下所述。

对于使用 Fargate 启动类型的任务，您将看到与您的任务关联的成本。

对于使用 EC2 启动类型的任务，任务不会具有与之关联的成本，但您可以使用 vCPU 和/或内存使用率来分配 Amazon EC2 实例的底层集群的成本。

您还可以使用 Amazon ECS 托管标签来标识每个任务所属的服务或集群。有关更多信息，请参阅 标记资源以便于计费 (p. 332)。

> **Important**
>
> 仅可查看在 2018 年 11 月 16 日或之后启动的任务的计量数据。在此日期之前启动的任务不会显示计量数据。

下面是您在使用 Cost Explorer 时可以按其对成本分配数据进行分类的一些字段示例：

- 集群名称
- 服务名称
- 资源标签
- 启动类型
- 区域
- 使用类型

有关创建 AWS 成本和使用率报告的更多信息，请参阅 AWS Billing and Cost Management 用户指南 中的 AWS 成本和使用率报告。

Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink)

您可以通过将 Amazon ECS 配置为使用接口 VPC 终端节点来改善 VPC 的安全状况。接口终端节点由 AWS PrivateLink 提供支持，您可以使用该技术通过私有 IP 地址私下访问 Amazon ECS API。PrivateLink 将 VPC 和 Amazon ECS 之间的所有网络流量限制在 Amazon 网络以内。您无需 Internet 网关、NAT 设备或虚拟专用网关。

不要求您配置 PrivateLink，但我们建议您这样做。有关 PrivateLink 和 VPC 终端节点的更多信息，请参阅通过 AWS PrivateLink 访问服务。

Amazon ECS VPC 终端节点注意事项

在为 Amazon ECS 设置接口 VPC 终端节点之前，请注意以下事项：

- 使用 Fargate 启动类型的任务无需 Amazon ECS 的接口 VPC 终端节点，但您可能需要 Amazon ECR 或 Amazon CloudWatch Logs 的接口 VPC 终端节点，如以下要点中所述。
  - 要允许您的任务从 Amazon ECR 拉取私有镜像，您必须为 Amazon ECR 创建接口 VPC 终端节点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南 中的接口 VPC 终端节点 (AWS PrivateLink)。

    Important

    如果您将 Amazon ECR 配置为使用接口 VPC 终端节点，则可以创建包含条件键的任务执行角色，以限制对特定 VPC 或 VPC 终端节点的访问。有关更多信息，请参阅通过接口终端节点拉取 Amazon ECR 映像的 Fargate 任务的可选 IAM 权限 (p. 409)。
  - 如果您的 VPC 没有 Internet 网关，并且您的任务使用 `awslogs` 日志驱动程序将日志信息发送到 CloudWatch Logs，则必须为 CloudWatch Logs 创建接口 VPC 终端节点。有关更多信息，请参阅 Amazon CloudWatch Logs User Guide 中的将 CloudWatch Logs 与接口 VPC 终端节点一起使用。
- 使用 EC2 启动类型的任务要求启动它们的容器实例至少运行 `1.25.1` 版本的 Amazon ECS 容器代理。有关更多信息，请参阅Amazon ECS 容器代理版本 (p. 207)。
- VPC 终端节点当前不支持跨区域请求。确保在计划向 Amazon ECS 发出 API 调用的同一区域中创建终端节点。
- VPC 终端节点仅通过 Amazon Route 53 支持 Amazon 提供的 DNS。如果您希望使用自己的 DNS，可以使用条件 DNS 转发。有关更多信息，请参阅 Amazon VPC 用户指南 中的 DHCP 选项集。
- 附加到 VPC 终端节点的安全组必须允许从 VPC 的私有子网通过端口 443 进行传入连接。
- 当前不支持通过将终端节点策略附加到 VPC 终端节点来控制对 Amazon ECS 的访问。默认情况下，允许通过接口终端节点对服务进行完全访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的使用 VPC 终端节点控制对服务的访问。

为 Amazon ECS 创建 VPC 终端节点

要为 Amazon ECS 服务创建 VPC 终端节点，请使用 Amazon VPC 用户指南 中的创建接口终端节点过程来创建以下终端节点。如果您的 VPC 中当前有容器实例，则应按其列出的顺序创建终端节点。如果您计划在创建 VPC 终端节点后创建容器实例，则顺序无关紧要。

- `com.amazonaws.`*`region`*`.ecs-agent`

- `com.amazonaws.region.ecs-telemetry`
- `com.amazonaws.region.ecs`

> **Note**
>
> *##* 表示 Amazon ECS 支持的 AWS 区域的区域标识符，如美国东部（俄亥俄）区域的 `us-east-2`。

如果您当前有使用 EC2 启动类型的任务，则在创建 VPC 终端节点后，每个容器实例都需要采用新的配置。为实现此目的，您必须重新启动每个容器实例或重新启动每个容器实例上的 Amazon ECS 容器代理。要重新启动容器代理，请执行以下操作。

重新启动 Amazon ECS 容器代理

1. 通过 SSH 登录到容器实例。有关更多信息，请参阅连接到您的容器实例 (p. 180)。
2. 停止容器代理。

```
sudo docker stop ecs-agent
```

3. 启动容器代理。

```
sudo docker start ecs-agent
```

在创建 VPC 终端节点并在每个容器实例上重新启动 Amazon ECS 容器代理后，所有新启动的任务都将采用新配置。

Amazon ECR 存储库

Amazon ECR 是一项托管 AWS Docker 注册表服务。客户可以使用熟悉的 Docker CLI 推送、拉取和管理映像。Amazon ECR 提供了安全、可扩展且可靠的注册表。Amazon ECR 通过使用 AWS IAM 支持具有基于资源的权限的私有 Docker 存储库，以便特定用户或 Amazon EC2 实例可以访问存储库和映像。开发人员可以使用 Docker CLI 编写和管理映像。

有关如何创建存储库、从 Amazon ECR 推送和拉取映像以及对存储库设置访问控制的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

在 Amazon ECS 中使用 Amazon ECR 映像

您可以将 ECR 映像与 Amazon ECS 结合使用，但需要满足以下先决条件：

- 您的容器实例必须至少使用版本 1.7.0 的 Amazon ECS 容器代理。最新版本的经 Amazon ECS –优化的 AMI 在任务定义中支持 ECR 映像。有关更多信息（包括最新的经 Amazon ECS– 优化的 AMI ID），请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器代理版本 (p. 207)。
- 您用于容器实例的 Amazon ECS 容器实例角色 (`ecsInstanceRole`) 必须拥有 Amazon ECR 的以下 IAM 策略权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

  如果您对容器实例使用 `AmazonEC2ContainerServiceforEC2Role` 托管策略，则您的角色将具有适当的权限。要检查您的角色是否支持 Amazon ECR，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器实例 IAM 角色 (p. 404)。
- 在 ECS 任务定义中，确保对 ECR 映像使用完整的 `registry/repository:tag` 命名。例如，*aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest*。

监控 Amazon ECS

您可以使用 Amazon CloudWatch 监控您的 Amazon ECS 资源，此工具可从 Amazon ECS 收集原始数据，并将数据处理为易读的近乎实时的指标。这些统计数据会保存两周，从而使您能够访问历史信息，并能够更好地了解集群或服务的执行情况。Amazon ECS 指标数据以 1 分钟为间隔自动发送到 CloudWatch。有关 CloudWatch 的更多信息，请参阅 Amazon CloudWatch 用户指南。

监控是保持 Amazon ECS 和您的 AWS 解决方案的可靠性、可用性和性能的重要方面。您应从 AWS 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。不过，在开始监控 Amazon ECS; 之前，您应制定一个监控计划并在计划中回答下列问题：

- 您的监控目标是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

可用的指标将取决于集群中任务和服务的启动类型。如果您为服务使用 Fargate 启动类型，则会提供 CPU 和内存利用率指标以帮助监控您的服务。对于 Amazon EC2 启动类型，您将拥有并需要监控构成底层基础设施的 EC2 实例。其他 CPU 以及内存预留和利用率指标可在集群、服务和任务级别上使用。

下一步，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 Amazon ECS 性能的基准。在监控 Amazon ECS 时，存储历史监控数据，以便将此数据与当前性能数据进行比较，确定正常性能模式和性能异常，并设计解决问题的方法。

要建立基准，您至少应监控以下各项：

- 您的 Amazon ECS 集群的 CPU 与内存预留和利用率指标
- 您的 Amazon ECS 服务的 CPU 和内存利用率指标

主题

- 监控工具 (p. 340)
- Amazon ECS CloudWatch 指标 (p. 341)
- CloudWatch Events 的 Amazon ECS 事件流 (p. 354)
- Amazon ECS CloudWatch Container Insights (p. 363)
- 使用 Amazon ECS 记录 AWS CloudTrail API 调用 (p. 365)

监控工具

AWS 提供各种可以用来监控 Amazon ECS 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

您可以使用以下自动化监控工具来监控 Amazon ECS 并在出现错误时报告：

- Amazon CloudWatch 警报 – 按您指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。操作是发送通知到 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略。CloudWatch 警报将不会仅因为其处于特定状态而调用操作；该状

态必须已改变并在指定的若干个时间段内保持不变。有关更多信息，请参阅Amazon ECS CloudWatch 指标 (p. 341)。

对于具有使用 EC2 启动类型的任务或服务的集群，您可以使用 CloudWatch 警报来根据 CloudWatch 指标（例如集群内存预留）扩展和缩减容器实例。有关更多信息，请参阅 教程：通过 CloudWatch 警报扩展容器实例 (p. 349)。

- Amazon CloudWatch Logs – 通过在任务定义中指定 `awslogs` 日志驱动程序，监控、存储和访问来自 Amazon ECS 任务中的容器的日志文件。这是访问使用 Fargate 启动类型的任务的日志的唯一受支持方法，但也适用于使用 EC2 启动类型的任务。有关更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。

  您也可以从 Amazon ECS 容器实例监控、存储和访问操作系统及 Amazon ECS 容器代理日志文件。这种访问日志的方法可以用于使用 EC2 启动类型的容器。有关更多信息，请参阅 将 CloudWatch Logs 用于容器实例 (p. 181)。
- Amazon CloudWatch Events – 匹配事件并将事件传送到一个或多个目标函数或流来进行更改、捕获状态信息和采取纠正措施。有关更多信息，请参阅本指南中的CloudWatch Events 的 Amazon ECS 事件流 (p. 354)和 Amazon CloudWatch Events 用户指南 中的什么是 Amazon CloudWatch Events？。
- AWS CloudTrail 日志监控 – 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 CloudWatch Logs 对它们进行实时监控，使用 Java 编写日志处理应用程序，以及验证日志文件是否在由 CloudTrail 传送后未发生更改。有关更多信息，请参阅本指南中的使用 Amazon ECS 记录 AWS CloudTrail API 调用 (p. 365)和 AWS CloudTrail User Guide 中的使用 CloudTrail 日志文件。

手动监控工具

监控 Amazon ECS 时的另一个重要环节是手动监控 CloudWatch 警报未涵盖的项目。CloudWatch、Trusted Advisor 和其他 AWS 控制台控制面板提供 AWS 环境状态的概览视图。建议您也可以查看容器实例上的日志文件以及任务中的容器。

- CloudWatch 主页：
  - 当前警报和状态
  - 警报和资源的图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务。
  - 绘制指标数据图，以排除问题并弄清楚趋势。
  - 搜索并浏览您所有的 AWS 资源指标。
  - 创建和编辑警报以接收有关问题的通知。
- AWS Trusted Advisor 可以帮助您监控 AWS 资源以提高性能、可靠性、安全性和成本效益。四个 Trusted Advisor 检查可供所有用户使用；超过 50 个检查可供具有“商业”或“企业”支持计划的用户使用。有关更多信息，请参阅 AWS Trusted Advisor。

Amazon ECS CloudWatch 指标

您可以使用 Amazon CloudWatch 监控您的 Amazon ECS 资源，此工具可从 Amazon ECS 收集原始数据，并将数据处理为易读的近乎实时的指标。这些统计数据会保存两周，从而使您能够访问历史信息，并能够更好地了解集群或服务的执行情况。Amazon ECS 指标数据以 1 分钟为间隔自动发送到 CloudWatch。有关 CloudWatch 的更多信息，请参阅 Amazon CloudWatch 用户指南。

主题

- 启用 CloudWatch 指标 (p. 342)
- 可用指标和维度 (p. 342)

- 集群预留 (p. 345)
- 集群利用率 (p. 346)
- 服务利用率 (p. 347)
- 服务 RUNNING 任务计数 (p. 347)
- 查看 Amazon ECS 指标 (p. 348)
- 教程：通过 CloudWatch 警报扩展容器实例 (p. 349)

启用 CloudWatch 指标

自动为 CloudWatch CPU 和内存利用率指标启用使用 Fargate 启动类型的任意 Amazon ECS 服务，因此您无需执行任何手动步骤。

对于使用 EC2 启动类型的任何 Amazon ECS 任务或服务，您的 Amazon ECS 容器实例都需要 1.4.0 版或更高版本的容器代理才能启用 CloudWatch 指标。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

如果您要手动启动代理（例如，如果您未对容器实例使用 Amazon ECS-optimized AMI），请参阅手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI）(p. 215)。

您的 Amazon ECS 容器实例还需要对您用来启动容器实例的 IAM 角色的 `ecs:StartTelemetrySession` 权限。如果您在 CloudWatch 指标可用于 Amazon ECS 之前已创建您的 Amazon ECS 容器实例，则可能需要添加此权限。有关检查您的 Amazon ECS 容器实例角色并为容器实例附加托管 IAM 策略的信息，请参阅在 IAM 控制台中检查 `ecsInstanceRole` (p. 405)。

> **Note**
>
> 您可以通过在 Amazon ECS 容器代理配置中设置 `ECS_DISABLE_METRICS=true` 来禁用 CloudWatch 指标收集。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。

可用指标和维度

以下部分列出了 Amazon ECS 发送到 Amazon CloudWatch 的指标和维度。

Amazon ECS 指标

Amazon ECS 为您提供了用来监控资源的指标。您可以将集群内的 CPU 和内存预留以及使用率作为一个整体进行监控，并监控集群中的各个服务的 CPU 和内存使用率。对于您的 GPU 工作负载，您可以度量集群中的 GPU 预留。

可用的指标将取决于集群中任务和服务的启动类型。如果您对服务使用 Fargate 启动类型，则会提供 CPU 和内存利用率指标以帮助监控您的服务。对于 EC2 启动类型，您将拥有并需要监控构成底层基础设施的 Amazon EC2 实例。因此，其他 CPU、内存和 GPU 预留以及 CPU 和内存利用率指标可在集群、服务和任务级别上使用。

Amazon ECS 每分钟向 CloudWatch 发送以下指标。当 Amazon ECS 收集指标时，每分钟会收集多个数据点。然后将它们聚合成一个数据点，并将数据发送到 CloudWatch。因此，在 CloudWatch 中，一个样本数实际上是每分钟内聚合的多个数据点。

`AWS/ECS` 命名空间包括以下指标。

指标	说明
`CPUReservation`	集群中正在运行的任务所预留的 CPU 单位的百分比。 集群 CPU 预留（此指标仅可按 `ClusterName` 进行筛选）是通过将集群上的 Amazon ECS 任务所预留的总

指标	说明
	CPU 单位数除以为集群中的所有容器实例注册的总 CPU 单位数计算得到的。仅处于 `ACTIVE` 或 `DRAINING` 状态的容器实例将影响 CPU 预留指标。此指标仅用于使用 EC2 启动类型 的任务。 有效维度：`ClusterName`。 有效统计数据：平均值、最小值、最大值、总计和样本数。最有用的统计数据是平均值。 单位：百分比。
`CPUUtilization`	集群或服务中使用的 CPU 单位的百分比。 集群 CPU 使用率（按 `ClusterName` 而不是 `ServiceName` 进行筛选的指标）是通过将集群上的 Amazon ECS 任务所使用的总 CPU 单位数除以为集群中的所有容器实例注册的总 CPU 单位数计算得到的。仅处于 `ACTIVE` 或 `DRAINING` 状态的容器实例将影响 CPU 利用率指标。集群 CPU 利用率指标仅用于使用 EC2 启动类型的任务。 服务 CPU 使用率（按 `ClusterName` 和 `ServiceName` 进行筛选的指标）是通过将属于服务的任务所使用的总 CPU 单位数除以为属于服务的任务预留的总 CPU 单位数计算得到的。服务 CPU 利用率指标用于使用 Fargate 和 EC2 启动类型的任务。 有效维度：`ClusterName`、`ServiceName`。 有效统计数据：平均值、最小值、最大值、总计和样本数。最有用的统计数据是平均值。 单位：百分比。
`MemoryReservation`	集群中正在运行的任务所预留的内存的百分比。 集群内存预留（此指标仅可按 `ClusterName` 进行筛选）是通过将集群上的 Amazon ECS 任务所预留的总内存量除以为集群中的所有容器实例注册的总内存量计算得到的。仅处于 `ACTIVE` 或 `DRAINING` 状态的容器实例将影响内存预留指标。此指标仅用于使用 EC2 启动类型的任务。 有效维度：`ClusterName`。 有效统计数据：平均值、最小值、最大值、总计和样本数。最有用的统计数据是平均值。 单位：百分比。

指标	说明
MemoryUtilization	集群或服务中使用的内存的百分比。 集群内存使用率（按 ClusterName 而不是 ServiceName 进行筛选的指标）是通过将集群上的 Amazon ECS 任务所使用的总内存量除以为集群中的所有容器实例注册的总内存量计算得到的。仅处于 ACTIVE 或 DRAINING 状态的容器实例将影响内存利用率指标。集群内存利用率指标仅用于使用 EC2 启动类型的任务。 服务内存使用率（按 ClusterName 和 ServiceName 进行筛选的指标）是通过将属于服务的任务所使用的总内存量除以为属于服务的任务预留的总内存量计算得到的。服务内存利用率指标用于使用 Fargate 和 EC2 启动类型的任务。 有效维度：ClusterName、ServiceName。 有效统计数据：平均值、最小值、最大值、总计和样本数。最有用的统计数据是平均值。 单位：百分比。
GPUReservation	集群中正在运行的任务所预留的总可用 GPU 的百分比。 集群 GPU 预留是通过将集群上的 Amazon ECS 任务所预留的 GPU 数除以集群中所有已启用 GPU 的容器实例上可用的 GPU 总数计算得到的。仅处于 ACTIVE 或 DRAINING 状态的容器实例将影响 GPU 预留指标。 有效维度：ClusterName。 有效统计数据：平均值、最小值、最大值、总计和样本数。最有用的统计数据是平均值。 单位：百分比。

Note

如果您使用的是使用 EC2 启动类型的任务并且具有 Linux 容器实例，则 Amazon ECS 容器代理将依靠 Docker stats 指标收集在实例上运行的每个容器的 CPU 和内存数据。如果您使用版本早于 1.14.0 的 Amazon ECS 代理，Amazon ECS 会在向 CloudWatch 报告内存利用率时包括文件系统缓存使用率，此时您的 CloudWatch 图表显示的使用率会高于任务实际内存利用率。为修复这一问题，自 Amazon ECS 代理版本 1.14.0 起，Amazon ECS 容器代理会从内存利用率指标中排除文件系统缓存使用率。此更改不会影响容器的内存不足行为。

Amazon ECS 指标的维度

Amazon ECS 指标使用 AWS/ECS 命名空间并提供以下维度的指标。

维度	说明
ClusterName	此维度将筛选您为指定集群中的所有资源请求的数据。将按 ClusterName 筛选所有 Amazon ECS 指标。

维度	说明
ServiceName	此维度将筛选您为指定集群内的指定服务中的所有资源请求的数据。

集群预留

集群预留指标以所有 Amazon ECS 任务在某个集群上预留的 CPU、内存和 GPU 占已为该集群中每个活动容器实例注册的聚合 CPU、内存和 GPU 的百分比来度量。仅处于 ACTIVE 或 DRAINING 状态的容器实例将影响集群预留指标。此指标仅用于包含使用 EC2 启动类型的任务或服务的集群。包含使用 Fargate 启动类型的任务的集群不支持此指标。

```
                                  (Total CPU units reserved by tasks in cluster) x 100
Cluster CPU reservation =  ------------------------------------------------------------
                           (Total CPU units registered by container instances in cluster)
```

```
                                     (Total MiB of memory reserved by tasks in cluster x
 100)
Cluster memory reservation =
 ---------------------------------------------------------------------
                               (Total MiB of memory registered by container instances in
 cluster)
```

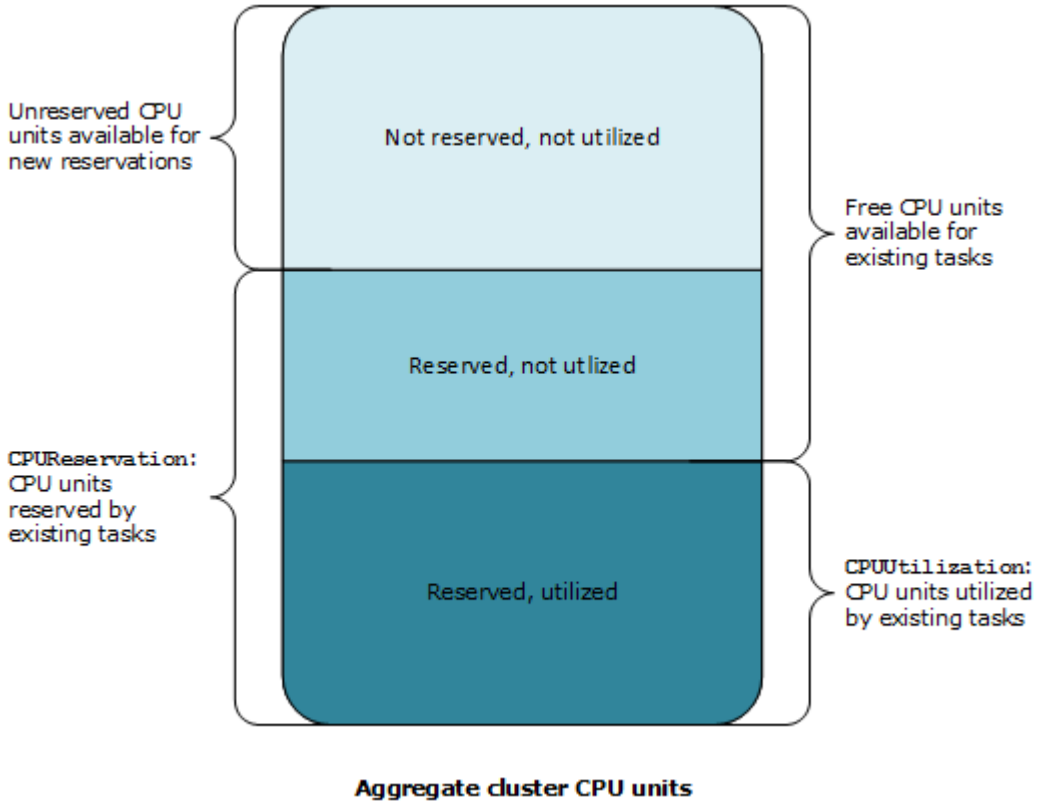
```
                                     (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ---------------------------------------------------------------------
                               (Total GPUs registered by container instances in cluster)
```

当您在某个集群中运行某项任务时，Amazon ECS 将分析该集群的任务定义并预留在该集群的容器定义中指定的聚合 CPU 单元数、内存 MiB 数和 GPU 数。Amazon ECS 每分钟都会计算当前为在该集群中运行的每个任务预留的 CPU 单元数、内存 MiB 数和 GPU 数。它还会计算为在该集群上运行的所有任务预留的 CPU、内存和 GPU 的总量，这些数字将以占该集群的注册资源总量的百分比的形式报告给 CloudWatch。如果您指定软限制 (memoryReservation)，它用于计算预留内存量。否则将使用硬限制 (memory)。有关软限制和硬限制的更多信息，请参阅任务定义参数。

例如，某个集群注册了两个活动容器实例：一个 c4.4xlarge 实例和一个 c4.large 实例。c4.4xlarge 实例在集群中注册了 16,384 个 CPU 单元和 30,158 MiB 内存。c4.large 实例注册了 2,048 个 CPU 单元和 3,768 MiB 内存。此集群的聚合资源为 18,432 个 CPU 单元和 33,926 MiB 内存。

如果某个任务定义预留了 1024 个 CPU 单元和 2048 MiB 内存，并且在此集群上启动了 10 个使用此任务定义的任务（当前没有运行其他任务），则总共将预留 10240 个 CPU 单元和 20480 MiB 内存。此信息将以集群的 55% CPU 预留量和 60% 内存预留量的形式报告给 CloudWatch。

下图显示了在某个集群中注册的 CPU 单元总数，以及它们的预留和利用率对现有任务和新任务放置的意义。下面的块（已预留，已使用）和中间的块（已预留，未使用）表示为集群上运行的现有任务预留的总 CPU 单元数（即 CPUReservation CloudWatch 指标）。下面的块表示正在运行的任务在集群上实际正在使用的预留 CPU 单元数（即 CPUUtilization CloudWatch 指标）。上面的块表示现有任务未预留的 CPU 单元数；这些 CPU 单元可用于新任务放置。现有任务如果对 CPU 资源的需求增加，也可以使用这些未预留 CPU 单元。有关更多信息，请参阅 cpu (p. 58) 任务定义参数文档。



集群利用率

集群利用率以所有 Amazon ECS 任务在某个集群上使用的 CPU 和内存占为该集群中每个活动容器实例注册的聚合 CPU 和内存的百分比的形式来度量。仅处于 `ACTIVE` 或 `DRAINING` 状态的容器实例将影响集群利用率指标。不支持 GPU 利用率指标，因为它无法过度使用 GPU。此指标仅用于包含使用 EC2 启动类型的任务或服务的集群。包含使用 Fargate 启动类型的任务的集群不支持此指标。

```
                                  (Total CPU units used by tasks in cluster) x 100
Cluster CPU utilization =  ------------------------------------------------------------
                           (Total CPU units registered by container instances in cluster)
```

```
                                     (Total MiB of memory used by tasks in cluster x 100)
Cluster memory utilization =
 ------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

每个容器实例上的 Amazon ECS 容器代理每分钟都会计算当前用于该容器实例上正在运行的每个任务的 CPU 单元数和内存 MiB 数，并将此信息报告回 Amazon ECS。它还会计算用于在该集群上运行的所有任务的 CPU 和内存的总量，并将这些数字以占该集群的总注册资源数的百分比的形式报告给 CloudWatch。

例如，某个集群注册了两个活动容器实例：一个 `c4.4xlarge` 实例和一个 `c4.large` 实例。`c4.4xlarge` 实例在集群中注册了 16,384 个 CPU 单元和 30,158 MiB 内存。`c4.large` 实例注册了 2,048 个 CPU 单元和 3,768 MiB 内存。此集群的聚合资源为 18,432 个 CPU 单元和 33,926 MiB 内存。

如果有 10 个任务正在此集群上运行，并且每个任务使用 1024 个 CPU 单元和 2048 MiB 内存，则该集群上总共使用了 10240 个 CPU 单元和 20480 MiB 内存。此信息将以集群的 55% CPU 利用率和 60% 内存利用率的形式报告给 CloudWatch。

服务利用率

服务利用率以属于集群上某个服务的 Amazon ECS 任务使用的 CPU 和内存占该服务的任务定义中指定的 CPU 和内存的百分比的形式来度量。 具有使用 EC2 和 Fargate 启动类型的任务的服务支持此指标。

```
                                    (Total CPU units used by tasks in service) x 100
Service CPU utilization =
 ---------------------------------------------------------------------------
                          (Total CPU units specified in task definition) x (number of
 tasks in service)
```

```
                                       (Total MiB of memory used by tasks in service) x
 100
Service memory utilization =
 ---------------------------------------------------------------------------------
                                 (Total MiB of memory specified in task definition) x (number
 of tasks in service)
```

每个容器实例上的 Amazon ECS 容器代理每分钟都会计算当前用于由该服务拥有且正在该容器实例上运行的每个任务的 CPU 单元数和内存 MiB 数，并将此信息报告回 Amazon ECS。它还将计算用于由该服务拥有且正在该集群上运行的所有任务的 CPU 和内存的总量，并将这些数字以占在该服务的任务定义中为该服务指定的总资源的百分比的形式报告给 CloudWatch。如果您指定软限制 (`memoryReservation`)，它用于计算预留内存量。否则将使用硬限制 (`memory`)。有关软限制和硬限制的更多信息，请参阅任务定义参数。

例如，某个服务的任务定义为其所有容器总共指定了 512 个 CPU 单元和 1,024 MiB 内存 (带硬限制 `memory` 参数)。该服务需要有 1 个正在运行的任务，并且该服务在一个包含 1 个 `c4.large` 容器实例 (具有 2,048 个 CPU 单元和 3,768 MiB 总内存) 的集群上运行，该集群上没有其他正在运行的任务。尽管该任务指定了 512 个 CPU 单元，但由于它是具有 2048 个 CPU 单元的容器实例上唯一一个正在运行的任务，它最多可以使用指定数量的四倍的资源 (2048/512)。但是，1024 MiB 的指定内存量是一个它无法超出的硬限制，因此在这种情况下，服务内存利用率不能超过 100%。

如果上述示例使用了软限制 `memoryReservation` 而不是硬限制 `memory` 参数，则该服务的任务可以根据需要使用超出指定的 1024MiB 的内存量。在这种情况下，该服务的内存利用率可能会超过 100%。

如果该任务在某个时间段内执行 CPU 密集型工作并且用尽了 2048 个可用 CPU 单元和 512 MiB 内存，则服务将报告 400% CPU 利用率和 50% 内存利用率。如果该任务处于空闲状态且使用了 128 个 CPU 单元和 128 MiB 内存，则服务将报告 25% CPU 利用率和 12.5% 内存利用率。

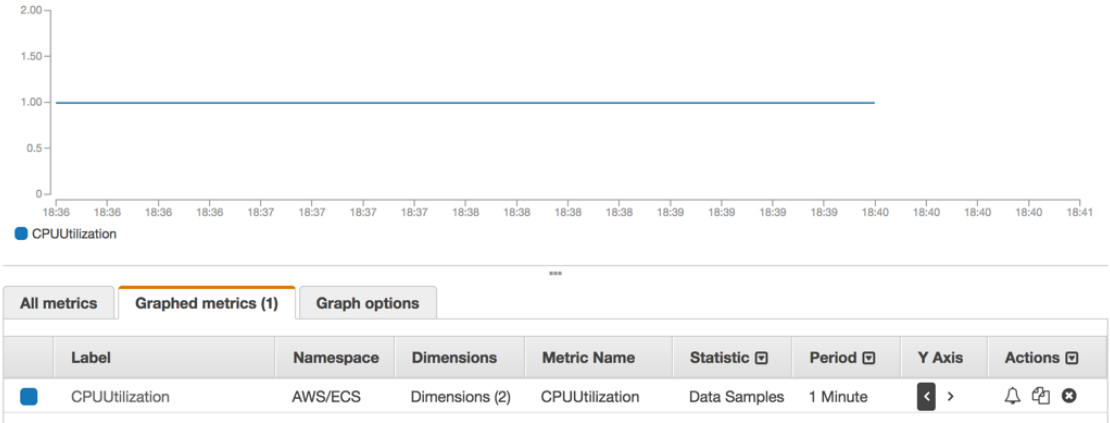
服务 RUNNING 任务计数

您可以使用 CloudWatch 指标查看您的服务中处于 `RUNNING` 状态的任务数。例如，您可以为此指标设置一个 CloudWatch 警报，用于在您的服务中运行的任务数下降至指定值以下时提醒您。

查看服务中正在运行的任务数

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格上，选择 Metrics (指标)。
3. 在 All metrics 选项卡上，选择 ECS。
4. 依次选择 ClusterName 和 ServiceName，然后选择与要在其中查看正在运行的任务的服务对应的任何指标（`CPUUtilization` 或 `MemoryUtilization`）。
5. 在绘成图表的指标选项卡上，将周期更改为 1 分钟，将统计数据更改为样本数。

   图中显示的值表示服务中的 `RUNNING` 任务的数量。



查看 Amazon ECS 指标

为 Amazon ECS 启用 CloudWatch 指标后，您可以在 Amazon ECS 控制台和 CloudWatch 控制台中查看这些指标。Amazon ECS 控制台提供了集群和服务指标的 24 小时最大值、最小值和平均值视图。CloudWatch 控制台提供了资源的精细的可自定义显示以及服务中正在运行的任务数。

主题

- 在 Amazon ECS 控制台上查看集群指标 (p. 348)
- 在 Amazon ECS 控制台上查看服务指标 (p. 348)
- 在 CloudWatch 控制台上查看 Amazon ECS 指标 (p. 349)

在 Amazon ECS 控制台上查看集群指标

集群和服务指标在 Amazon ECS 控制台上可用。为集群指标提供的视图显示了上一个 24 小时周期内的平均、最小和最大值，数据点每 5 分钟提供一次。有关集群指标的更多信息，请参阅集群预留 (p. 345)和集群利用率 (p. 346)。

在控制台上查看集群指标

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择要查看其指标的集群。
3. 在 Cluster: ***cluster-name*** (集群: <集群名称>) 页面上，选择 Metrics (指标)。

在 Amazon ECS 控制台上查看服务指标

Amazon ECS 服务 CPU 和内存利用率指标在 Amazon ECS 控制台上可用。为服务指标提供的视图显示了上一个 24 小时周期内的平均、最小和最大值，数据点每 5 分钟提供一次。有关更多信息，请参阅服务利用率 (p. 347)。

在控制台中查看服务指标

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择包含要查看其指标的服务的集群。
3. 在 Cluster: ***cluster-name*** (集群: <集群名称>) 页面上，选择 Services (服务)。
4. 选择要查看其指标的服务。

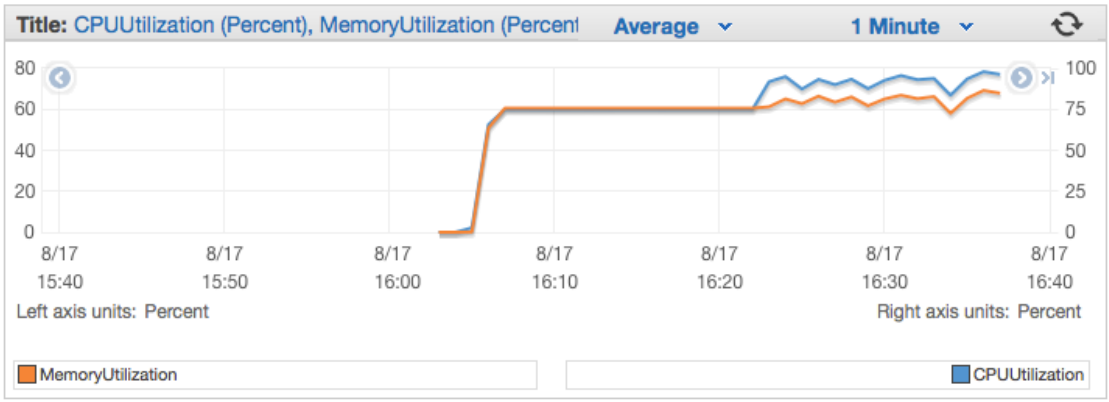
5. 在 Service: ***service-name*** (服务: <服务名称>) 页面上，选择 Metrics (指标)。

在 CloudWatch 控制台上查看 Amazon ECS 指标

Amazon ECS 集群和服务指标也可在 CloudWatch 控制台上查看。该控制台提供了 Amazon ECS 指标的最详细的视图，您可以根据自己的需求定制视图。您可以查看集群预留 (p. 345)、集群利用率 (p. 346)、服务利用率 (p. 347)和服务 `RUNNING` 任务计数 (p. 347)。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

在 CloudWatch 控制台中查看指标

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格的 Metrics (指标) 部分中，选择 ECS。
3. 选择要查看的指标。集群指标被限定为 ECS > ClusterName，服务利用率指标被限定为 ECS > ClusterName, ServiceName。以下示例显示了集群 CPU 和内存利用率。



教程：通过 CloudWatch 警报扩展容器实例

以下过程帮助您为 Amazon ECS 集群创建 Auto Scaling 组。Auto Scaling 组包含可使用 CloudWatch 警报扩展（和缩减）的容器实例。

根据您在集群中使用的 Amazon EC2 实例类型和每个集群中的容器实例数，您的任务在运行时可使用的资源数量是有限的。Amazon ECS 监控集群中可用的资源以便与计划程序一起来放置任务。如果您的集群缺少其中任何资源（例如内存），您最终将无法启动更多任务，直到您添加更多容器实例、减少服务中所需的任务数或停止集群中的一些正在运行的任务以释放受限制的资源。

在本教程中，您将使用 `MemoryReservation` 指标为集群创建一个 CloudWatch 警报。当集群的内存预留上升到 75% 以上（这意味着，集群中仅 25% 的内存可用于要预留的新任务）时，警报将触发 Auto Scaling 组添加另一个实例并为您的任务和服务提供更多资源。

先决条件

本指南假定您已为集群和服务启用 CloudWatch 指标。在集群和服务将指标发送到 CloudWatch 之前，指标不可用，并且您无法为不存在的指标创建 CloudWatch 警报。

您的 Amazon ECS 容器实例需要至少 1.4.0 版的容器代理才能启用 CloudWatch 指标。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。

您的 Amazon ECS 容器实例还需要对您用来启动容器实例的 IAM 角色的 `ecs:StartTelemetrySession` 权限。如果您在 CloudWatch 指标可用于 Amazon ECS 之前已创建您的 Amazon ECS 容器实例，则可能

需要添加此权限。有关检查您的 Amazon ECS 容器实例角色并为容器实例附加托管 IAM 策略的信息，请参阅在 IAM 控制台中检查 `ecsInstanceRole` (p. 405)。

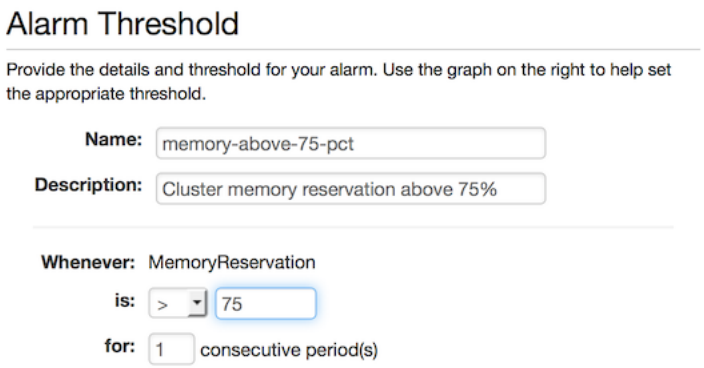
步骤 1：为指标创建 CloudWatch 警报

为集群和服务启用 CloudWatch 指标并且 CloudWatch 控制台中显示集群的指标后，您可以在这些指标上设置警报。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的创建 Amazon CloudWatch 警报。

对于本教程，您将在集群 `MemoryReservation` 指标上创建警报，以便在集群的内存预留高于 75% 时发出提醒。

在指标上创建 CloudWatch 警报

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航中，依次选择警报和创建警报。
3. 在 CloudWatch Metrics by Category 部分中，选择 ECS Metrics > ClusterName。
4. 在 Modify Alarm 页面上，选择默认集群的 `MemoryReservation` 指标，然后选择 Next。
5. 在 Alarm Threshold 部分中，为您的警报输入名称和描述。
   - 名称：`memory-above-75-pct`
   - 描述：`Cluster memory reservation above 75%`
6. 将阈值和时间段要求设置为大于 1 个周期的 75% 的 `MemoryReservation`。



7. （可选）配置在触发警报时发送的通知。如果您现在不想配置通知，也可以选择删除通知。
8. 选择 Create Alarm。现在，您可以使用此警报触发您的 Auto Scaling 组以在内存预留高于 75% 时添加容器实例。
9. （可选）您也可以创建另一个在内存预留低于 25% 时触发的警报，您可使用该警报来删除 Auto Scaling 组中的容器实例。

步骤 2：为 Auto Scaling 组创建启动配置

现在，您已启用 CloudWatch 指标，并且已创建基于这些指标之一的警报，您可以为集群创建一个启动配置和一个 Auto Scaling 组。有关更多信息和其他配置选项，请参阅 Amazon EC2 Auto Scaling 用户指南。

创建 Auto Scaling 启动配置

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格中，选择 Auto Scaling 组。
3. 在 Welcome to Auto Scaling (欢迎使用 Auto Scaling) 页面上，选择 Create Group (创建 Auto Scaling 组)。
4. 在创建 Auto Scaling 组页面上，选择 Create a new launch configuration (创建新启动配置)。

5. 在 Create Auto Scaling Group (创建 Auto Scaling 组) 向导的 Choose (选择 AMI) 步骤中，选择 Community AMIs (社区 AMI)。
6. 为 Auto Scaling 组选择经 ECS 优化的 AMI。

   要使用 Amazon ECS-optimized Amazon Linux 2 AMI，请在 Search community AMIs (搜索社区 AMI) 字段中键入 amzn2-ami-ecs 并按 Enter 键。选择 amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs AMI 旁边的选择。

   下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux 2 AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0dca97e7cde7be3b5	Launch instance
us-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0fac5486e4cff37f4	Launch instance
us-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0c6e63b58aac1043e	Launch instance
us-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0e5e051fd0b505db6	Launch instance
ap-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-02252d984c7e3595d	Launch instance
ap-northeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04a735b489d2a0320	Launch instance
ap-northeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0accbb5aa909be7b5	Launch instance
ap-south-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a8bf4e187339e2c1	Launch instance
ap-southeast-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-05c6d22d98f9747bc	Launch instance
ap-southeast-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-039bb4c3a7946ce19	Launch instance
ca-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-01c07ee95e77abba8	Launch instance
eu-central-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0650e7d86452db33b	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
eu-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-059aa04f0c253adcb	Launch instance
eu-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0ae254c8a2d3346a7	Launch instance
eu-west-2	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0de1dc478496a9e2b	Launch instance
eu-west-3	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0d260f3e5ccd06043	Launch instance
sa-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-038707d64e5b8e7ba	Launch instance
us-gov-east-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0a224902b35f8adca	Launch instance
us-gov-west-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-04c68165	Launch instance
cn-north-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance
cn-northwest-1	amzn2-ami-ecs-hvm-2.0.20190709-x86_64-ebs	ami-0b619dec66f0fe1eb	Launch instance

7. 在 Create Auto Scaling Group (创建 Auto Scaling 组) 向导的 Choose Instance Type (选择实例类型) 步骤中，为您的 Auto Scaling 组选择一个实例类型，然后选择 Next: Configure details (下一步: 配置详细信息)。
8. 在 Create Auto Scaling Group 向导的 Configure details 步骤中，输入以下信息。其他字段是可选的。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南 中的创建启动配置。
   - Name：为您的启动配置输入名称。
   - IAM role：为您的容器实例选择 `ecsInstanceRole`。如果您没有配置此角色，请参阅 Amazon ECS 容器实例 IAM 角色 (p. 404)。
   - IP Address Type (IP 地址类型)：选择您的容器实例的 IP 地址类型选项。要允许外部流量进入您的容器，请选择 Assign a public IP address to every instance (向每个实例分配公有 IP 地址)。
9. （可选）如果您有要与 Amazon EC2 用户数据一起传递到容器实例的配置信息，请选择高级详细信息并在用户数据字段中输入您的用户数据。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。
10. 选择 Next: Add Storage。
11. 在创建 Auto Scaling 组向导的添加存储步骤中，对您的实例进行所需的任何存储配置更改，然后选择下一步: 配置安全组。
12. 在创建 Auto Scaling 组向导的配置安全组步骤中，选择符合容器需求的现有安全组，或创建新的安全组并选择审核。
13. 查看启动配置并选择 Create launch configuration。

14. 选择用于通过 SSH 连接到实例的私有密钥，然后选择创建启动配置。继续使用新启动配置创建 Auto Scaling 组。

步骤 3：为集群创建 Auto Scaling 组

在启动配置完成后，请继续以下过程来创建使用启动配置的 Auto Scaling 组。

创建 Auto Scaling 组

1. 在创建 Auto Scaling 组向导的 Configure Auto Scaling group details (配置 Auto Scaling 组详细信息) 步骤中，输入以下信息，然后选择 Next: Configure scaling policies (下一步: 配置扩展策略)：
   - Group name: (组名:) 输入 Auto Scaling 组的名称。
   - Group size (组大小)：指定 Auto Scaling 组最初应拥有的容器实例数。
   - 网络：选择要在其中启动容器实例的 VPC。
   - 子网：选择要在其中启动容器实例的子网。对于高可用性集群，建议您启用区域中的所有子网。
2. 在 Create Auto Scaling Group 向导的 Configure scaling policies 步骤中，选择 Use scaling policies to adjust the capacity of this group。
3. 输入 Auto Scaling 组的最小容器实例数和最大容器实例数。
4. 在 Increase Group Size (增加组大小) 部分中，输入以下信息：
   - 执行策略的时间：选择您之前配置的 `memory-above-75-pct` CloudWatch 警报。
   - 请执行以下操作：输入要在触发警报时添加到集群的实例数。
5. 如果您配置了一个警报来触发组大小减小，请在 Decrease Group Size 部分中设置该警报，并指定触发该警报时删除的实例数。否则，请选择 Decrease Group Size (减小组大小) 部分右上角的 X 来折叠该部分。

   Note

   如果您将 Auto Scaling 组配置为删除容器实例，则任何正在已删除容器实例上运行的任务都将停止。在任务正作为服务的一部分运行时，如果所需资源（CPU、内存、端口）可用，Amazon ECS 将在另一个实例上重新启动这些任务。不过，手动启动的任务不会自动重新启动。
6. 依次选择审核和创建 Auto Scaling 组。

步骤 4：验证和测试 Auto Scaling 组

现在，您已创建 Auto Scaling 组，您应能够在 Amazon EC2 控制台 Instances (实例) 页面上看到实例正在启动。这些实例启动后也将注册到 Amazon ECS 集群中。

要测试您的 Auto Scaling 组是否已正确配置，请创建一些使用大量内存的任务并开始在集群中启动它们。您的集群在指定的时间段内超过了 CloudWatch 警报中的 75% 内存预留后，您应在 Amazon EC2 控制台中看到新的实例启动。

步骤 5：清理

完成本教程后，您可以选择在服务中为集群保留 Auto Scaling 组和 Amazon EC2 实例。但是，如果您没有主动使用这些资源，则应考虑将其清除以免您的账户产生不必要的费用。您可以删除 Auto Scaling 组以终止其中的 Amazon EC2 实例，但启动配置保持不变。您稍后可以使用该启动配置创建新的 Auto Scaling 组（如果您选择如此）。

删除 Auto Scaling 组

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。

2. 在左侧导航窗格中，选择 Auto Scaling 组。
3. 选择您之前创建的 Auto Scaling 组。
4. 依次选择 Actions 和 Delete。
5. 选择 Yes, Delete。

CloudWatch Events 的 Amazon ECS 事件流

您可以使用 CloudWatch Events 的 Amazon ECS 事件流来接收有关 Amazon ECS 集群的当前状态的近实时通知。如果您的任务使用 Fargate 启动类型，您可以看到您的任务的状态。 如果您的任务使用 EC2 启动类型，您可以看到容器实例的状态和这些容器实例上运行的所有任务的当前状态。

使用 CloudWatch Events，您可以基于 Amazon ECS 构建负责跨集群编排任务的自定义计划程序并近实时监视这些集群的状态。您无需计划和监控用于持续轮询 Amazon ECS 服务以了解状态更改的代码，而是使用任意 CloudWatch Events 目标（例如，AWS Lambda、Amazon Simple Queue Service、Amazon Simple Notification Service 和 Amazon Kinesis Data Streams）以异步方式处理 Amazon ECS 状态更改。

Amazon ECS 事件流确保每个事件至少传送一次。如果发送了重复事件，事件会提供足量信息来确定重复项。有关更多信息，请参阅处理事件 (p. 359)。

将对事件进行相关排序，以便您能够轻松告知与其他事件相关的某个事件何时发生。

主题

- Amazon ECS 事件 (p. 354)
- 处理事件 (p. 359)
- 教程：侦听 Amazon ECS CloudWatch Events (p. 360)
- 教程：发送任务已停止事件的 Amazon Simple Notification Service 警报 (p. 362)

Amazon ECS 事件

Amazon ECS 将两种类型的事件发送到 CloudWatch Events：容器实例事件和任务事件。只有您对任务使用 EC2 启动类型，才会发送容器实例事件。对于使用 Fargate 启动类型的任务，您将只收到任务状态事件。Amazon ECS 跟踪容器实例和任务的状态。如果任一资源发生更改，将触发事件。这些事件分类为容器实例状态更改事件或任务状态更改事件。以下部分中更详细地介绍了这些事件及可能原因。

> **Note**
>
> Amazon ECS 将来可能会增加其他事件类型、源和详细信息。如果您以编程方式对事件 JSON 数据反序列化，请确保应用程序已准备好处理未知属性，以避免在增加这些附加属性时出现问题。

在某些情况下，将为同一活动触发多个事件。例如，在容器实例上启动任务时，将为新任务触发任务状态更改事件。触发容器实例状态更改事件以说明容器实例上的可用资源（例如 CPU、内存和可用端口）的更改。同样，如果终止容器实例，将为容器实例、容器代理连接状态以及在容器实例上运行的每个任务触发事件。

事件包含两个 `version` 字段；一个字段在事件的主体中，一个字段在事件的 `detail` 对象中。

- 对于所有事件，事件主体中的版本设为 `0`。有关 CloudWatch Events 参数的更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的事件和事件模式。
- 事件的 `detail` 对象中的版本描述了关联资源的版本。当资源状态发生更改时，此版本会递增。由于可多次发送事件，因此您可以使用该字段来确定重复事件（它们在 `detail` 对象中具有同一版本）。如果您使用 CloudWatch Events 复制 Amazon ECS 容器实例和任务状态，则可比较 Amazon ECS API 所报告的资源版本与 CloudWatch Events 中报告的资源版本（在 `detail` 对象中），以便验证事件流中的版本是否为最新版本。

主题

- 容器实例状态更改事件 (p. 355)
- 任务状态更改事件 (p. 357)

容器实例状态更改事件

以下方案将触发容器实例状态更改事件：

您调用 `StartTask`、`RunTask` 或 `StopTask` API 操作（直接调用，或者通过 AWS 管理控制台或开发工具包调用）。

在容器实例上放置或停止任务将修改容器实例上的可用资源（例如 CPU、内存和可用端口）。

Amazon ECS 服务计划程序启动或停止任务。

在容器实例上放置或停止任务将修改容器实例上的可用资源（例如 CPU、内存和可用端口）。

Amazon ECS 容器代理为具有所需的 `RUNNING` 状态的任务调用带 `STOPPED` 状态的 `SubmitTaskStateChange` API 操作。

Amazon ECS 容器代理监控容器实例上的任务状态，并报告任何状态更改。如果应为 `RUNNING` 的任务转换为 `STOPPED`，则代理将释放分配给已停止任务的资源（例如 CPU、内存和可用端口）。

您可以使用 `DeregisterContainerInstance` API 操作（直接调用，或者通过 AWS 管理控制台或开发工具包调用）取消注册容器实例。

取消注册容器实例将更改容器实例的状态以及 Amazon ECS 容器代理的连接状态。

任务已在 EC2 实例停止后停止。

在停止容器实例时，该实例上运行的任务会转换为 `STOPPED` 状态。

Amazon ECS 容器代理首次注册容器实例。

当 Amazon ECS 容器代理首次注册容器实例（在启动时注册，或者在首次运行时手动注册）时，将为实例创建状态更改事件。

Amazon ECS 容器代理与 Amazon ECS 连接或断开连接。

当 Amazon ECS 容器代理与 Amazon ECS 后端连接或断开连接时，将更改容器实例的 `agentConnected` 状态。

> **Note**
>
> Amazon ECS 容器代理在正常运行过程中会定期断开连接并重新连接（每小时多次），因此应会出现代理连接事件，这些事件并不表示容器代理或您的容器实例出现问题。

升级实例上的 Amazon ECS 容器代理。

容器实例详细信息包含容器代理版本的对象。如果您升级代理，此版本信息将发生更改并且将触发事件。

Example 容器实例状态更改事件

容器实例状态更改事件以下面的格式传送（下面的 `detail` 部分类似于从 Amazon Elastic Container Service API Reference 中的 DescribeContainerInstances API 操作返回的 ContainerInstance 对象）。有关 CloudWatch Events 参数的更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的事件和事件模式。

```
{
  "version": "0",
  "id": "8952ba83-7be2-4ab5-9c32-6687532d15a2",
  "detail-type": "ECS Container Instance State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
```

```
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315"
  ],
  "detail": {
    "agentConnected": true,
    "attributes": [
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
      },
      {
        "name": "com.amazonaws.ecs.capability.privileged-container"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      }
    ],
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "ec2InstanceId": "i-f3a8506b",
    "registeredResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 2048
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 3767
      },
```

```
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "remainingResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 1988
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "status": "ACTIVE",
    "version": 14801,
    "versionInfo": {
      "agentHash": "aebcbca",
      "agentVersion": "1.13.0",
      "dockerVersion": "DockerVersion: 1.11.2"
    },
    "updatedAt": "2016-12-06T16:41:06.991Z"
  }
}
```


任务状态更改事件

以下方案将触发任务状态更改事件：

您调用 `StartTask`、`RunTask` 或 `StopTask` API 操作（直接调用，或者通过 AWS 管理控制台、AWS CLI 或开发工具包调用）。

启动或停止任务将创建新的任务资源或修改现有任务资源的状态。

Amazon ECS 服务计划程序启动或停止任务。

启动或停止任务将创建新的任务资源或修改现有任务资源的状态。

Amazon ECS 容器代理调用 `SubmitTaskStateChange` API 操作。

Amazon ECS 容器代理监控您的容器实例上的的状态并报告任何状态更改（例如，从 `PENDING` 更改为 `RUNNING`，或者从 `RUNNING` 更改为 `STOPPED`）。

您可以使用 `DeregisterContainerInstance` API 操作和 `force` 标志（直接调用，或者通过 AWS 管理控制台或开发工具包调用）强制取消注册基础容器实例。

取消注册容器实例将更改容器实例的状态以及 Amazon ECS 容器代理的连接状态。如果任务正在容器实例上运行，则必须设置 `force` 标志以允许取消注册。这将停止实例上的所有任务。

停止或终止基础容器实例。

在停止或终止容器实例时，该实例上运行的任务会转换为 `STOPPED` 状态。

任务中的容器状态发生更改。

Amazon ECS 容器代理监控任务中的容器状态。例如，如果在任务中运行的容器停止，则此容器状态更改将触发事件。

Example 任务状态更改事件

任务状态更改事件以下面的格式传送（下面的 `detail` 部分类似于从 Amazon Elastic Container Service API Reference 中的 DescribeTasks API 操作返回的 Task 对象）。有关 CloudWatch Events 参数的更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的事件和事件模式。

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:task/b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
  ],
  "detail": {
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:111122223333:container/3305bea1-
bd16-4217-803d-3e0482170a17",
        "exitCode": 0,
        "lastStatus": "STOPPED",
        "name": "xray",
        "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
      }
    ],
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "RUNNING",
    "group": "task-group",
    "lastStatus": "RUNNING",
    "overrides": {
      "containerOverrides": [
        {
```

```
          "name": "xray"
        }
      ]
    },
    "startedAt": "2016-12-06T16:41:06.8Z",
    "startedBy": "ecs-svc/9223370556150183303",
    "updatedAt": "2016-12-06T16:41:06.975Z",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6f1cebef",
    "taskDefinitionArn": "arn:aws:ecs:us-east-1:111122223333:task-definition/xray:2",
    "version": 4
  }
}
```

处理事件

Amazon ECS 至少发送事件一次。这意味着，您可能收到给定事件的多个副本。此外，无法按事件的发生顺序将事件传送到事件侦听器。

要启用适当的事件排序，每个事件的 `detail` 部分均包含一个 `version` 属性。版本属性数值更大的事件应视为在版本号更小的事件之后发生。具有匹配版本号的事件可视为重复事件。

示例：在 AWS Lambda 函数中处理事件

以下示例显示了一个用 Python 2.7 编写的 Lambda 函数，此函数可同时捕获任务和容器实例状态更改事件，并将这些事件保存到两个 Amazon DynamoDB 表之一：

- ECSCtrInstanceState：存储容器实例的最新状态。表 ID 是容器实例的 `containerInstanceArn` 值。
- ECSTaskState：存储任务的最新状态。表 ID 是任务的 `taskArn` 值。

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print(json.dumps(event))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type of:
 aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    elif event["detail-type"] == "ECS Container Instance State Change":
        table_name = "ECSCtrInstanceState"
        id_name =  "containerInstanceArn"
        event_id = event["detail"]["containerInstanceArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting without
 saving event.")

    new_record["cw_version"] = event["version"]
```

```
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling")
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event - ignoring")
    else:
        print("Saving new event - ID " + event_id)

        table.put_item(
            Item=new_record
        )
```

教程：侦听 Amazon ECS CloudWatch Events

在本教程中，您设置一个简单的 AWS Lambda 函数，该函数会侦听 Amazon ECS 任务事件并将这些事件写出到 CloudWatch Logs 日志流。

先决条件：设置测试集群

如果您没有要从中捕获事件的正在运行的集群，请执行 开始使用 Amazon ECS (p. 19)中的步骤来创建一个集群。在本教程结束时，您可在此集群上运行任务来测试您是否已正确配置 Lambda 函数。

步骤 1：创建 Lambda 函数

在此过程中，您将创建一个简单的 Lambda 函数来充当 Amazon ECS 事件流消息的目标。

1. 通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 选择 Create function (创建函数)。
3. 在 Author from scratch 屏幕上，执行以下操作：
   a. 对于名称，输入一个值。
   b. 对于 Runtime (运行时)，选择 Python 2.7。
   c. 对于 Role (角色)，选择 Create a new role with basic Lambda permissions (创建具有基本 Lambda 权限的新角色)。

4. 选择 Create function (创建函数)。
5. 在 Function code 部分中，编辑示例代码以匹配以下示例：

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    print('Here is the event:')
    print(json.dumps(event))
```

   这是一个简单的 Python 2.7 函数，可输出由 Amazon ECS 发送的事件。如果一切配置正确，则在本教程结束时，您将在与此 Lambda 函数关联的 CloudWatch Logs 日志流中看到事件详细信息。
6. 选择 Save。

步骤 2：注册事件规则

接下来，您创建一个 CloudWatch Events 事件规则，该规则可捕获来自 Amazon ECS 集群的任务事件。该规则捕获来自定义该规则的账户中的所有集群的所有事件。任务消息本身包含有关事件源的信息（包括事件源所在的集群），可使用这些信息以编程方式对事件进行筛选和排序。

> **Note**
>
> 在使用 AWS 管理控制台创建事件规则时，控制台会自动添加所需的 IAM 权限以便向 CloudWatch Events 授予调用 Lambda 函数所需的权限。如果您使用 AWS CLI 创建事件规则，则需要明确授予此权限。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的事件和事件模式。

将事件路由到 Lambda 函数

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格上，依次选择 Events (事件)、Rules (规则)、Create rule (创建规则)。
3. 对于 Event Source (事件源)，选择 ECS 作为事件源。默认情况下，规则将应用于您的所有 Amazon ECS 组的所有 Amazon ECS 事件。或者，您可以选择特定的事件或特定的 Amazon ECS 组。
4. 对于 Targets (目标)，选择 Add target (添加目标)，对于 Target type (目标类型)，选择 Lambda function (Lambda 函数)，然后选择您的 Lambda 函数。
5. 选择 Configure details。
6. 对于 Rule definition，键入规则的名称和说明，然后选择 Create rule。

步骤 3：测试您的规则

最后，您创建一个 CloudWatch Events 事件规则，该规则可捕获来自 Amazon ECS 集群的任务事件。该规则捕获来自定义该规则的账户中的所有集群的所有事件。任务消息本身包含有关事件源的信息（包括事件源所在的集群），可使用这些信息以编程方式对事件进行筛选和排序。

测试您的规则

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择 Clusters 和 default。
3. 在 Cluster : default 屏幕上，选择 Tasks 和 Run new Task。
4. 对于 Task Definition，选择最新版本的 console-sample-app-static，然后选择 Run Task。
5. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。

6. 在导航窗格中，选择 Logs (日志)，然后选择 Lambda 函数的日志组（例如，/aws/lambda/*my-function*）。
7. 选择日志流以查看事件数据。

教程：发送任务已停止事件的 Amazon Simple Notification Service 警报

在本教程中，您配置一个 CloudWatch Events 事件规则，此规则仅捕获其中的任务因某个主要容器终止而停止的任务事件。事件仅将具有特定 `stoppedReason` 属性的任务事件发送到指定的 Amazon SNS 主题。

先决条件：设置测试集群

如果您没有要从中捕获事件的正在运行的集群，请执行 开始使用 Amazon ECS (p. 19)中的步骤来创建一个集群。在本教程结束时，您在此集群上运行一个任务来测试您是否已正确配置 Amazon SNS 主题和 CloudWatch Events 事件规则。

步骤 1：创建并订阅 Amazon SNS 主题

在本教程中，您配置一个 Amazon SNS 主题来充当新事件规则的事件目标。

创建 Amazon SNS 主题

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 选择 Topics 和 Create new topic。
3. 在 Create new topic 窗口上，为 Topic name 输入 TaskStoppedAlert，并选择 Create topic。
4. 在 Topics 窗口上，选择刚刚创建的主题。在 Topic details: TaskStoppedAlert 屏幕上，选择 Create subscription。
5. 在 Create Subscription 窗口上，为 Protocol 选择 Email。对于 Endpoint，输入您当前有权访问的电子邮件地址，然后选择 Create subscription。
6. 检查您的电子邮件账户，并等待接收订阅确认电子邮件。在收到此电子邮件后，选择 Confirm subscription。

步骤 2：注册事件规则

接下来，您注册一个事件规则，此规则仅捕获具有已停止容器的任务的任务已停止事件。

创建事件规则

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格上，选择 Events 和 Create rule。
3. 选择 Show advanced options 和 edit。
4. 对于 Build a pattern that selects events for processing by your targets，将现有文本替换为以下文本：

```
{
"source": [
  "aws.ecs"
],
"detail-type": [
  "ECS Task State Change"
],
"detail": {
  "lastStatus": [
```

```
    "STOPPED"
  ],
  "stoppedReason" : [
    "Essential container in task exited"
  ]
}
}
```

此代码定义一个 CloudWatch Events 事件规则，此规则匹配其中的 `lastStatus` 和 `stoppedReason` 字段均匹配所指示值的任何事件。有关事件模式的更多信息，请参阅 Amazon CloudWatch 用户指南 中的事件和事件模式。

5. 对于 Targets，选择 Add target。对于 Target type，选择 SNS topic，然后选择 TaskStoppedAlert。
6. 选择 Configure details。
7. 对于 Rule definition，键入规则的名称和说明，然后选择 Create rule。

步骤 3：测试您的规则

要测试您的规则，可尝试运行在启动后快速退出的任务。如果您的事件规则配置正确，您将在几分钟内收到包含事件文本的电子邮件。

测试规则

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择 Task Definitions 和 Create new Task Definition。
3. 对于 Task Definition Name，键入 WordPressFailure 并选择 Add Container。
4. 对于 Container name，键入 Wordpress；对于 Image，键入 wordpress；对于 Maximum memory (MB)，键入 128。
5. 选择 Add 和 Create。
6. 在 Task Definition 屏幕上，选择 Actions 和 Run Task。
7. 对于 Cluster (集群)，选择 default (默认)。选择 Run Task (运行任务)。
8. 在集群的 Tasks 选项卡上，定期选择刷新图标，直到您不再看到任务正在运行。要验证任务是否已停止，请为 Desired task status (预期任务状态) 选择 Stopped (已停止)。
9. 查看您的电子邮件以确认您已收到已停止通知的电子邮件警报。

Amazon ECS CloudWatch Container Insights

> CloudWatch Container Insights is in open preview. The preview is open to all AWS accounts and you do not need to request access. Features may be added or changed before announcing General Availability. Don’t hesitate to contact us with any feedback or let us know if you would like to be informed when updates are made by emailing us at containerinsightsfeedback@amazon.com

CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合和汇总指标与日志。指标包括资源的使用率，如 CPU、内存、磁盘和网络。CloudWatch 自动控制面板中提供了指标。有关 Amazon ECS Container Insights 指标的完整列表，请参阅 Amazon CloudWatch 用户指南 中的 Amazon ECS Container Insights 指标。

运行数据是作为性能日志事件收集的。它们是使用结构化 JSON 架构的条目，该架构允许批量提取和存储高基数数据。从该数据中，CloudWatch 在集群和服务级别创建更高级别的聚合指标以作为 CloudWatch 指标。有关更多信息，请参阅 Amazon CloudWatch 用户指南 中的 Amazon ECS 的 Container Insights 结构化日志。

> **Important**
>
> 提供 CloudWatch Container Insights 需要额外的成本。有关在不产生额外成本的情况下提供的默认监控指标的信息，请参阅 Amazon ECS CloudWatch 指标 (p. 341)。

使用已启用 Container Insights 的集群

可以选择使用 `containerInsights` 账户设置为所有集群启用 Container Insights，也可以在集群创建期间使用集群设置在单个集群上启用 Container Insights。可通过 Amazon ECS 控制台和 AWS CLI 选择使用 Container Insights 账户设置。有关创建 Amazon ECS 集群的更多信息，请参阅 创建集群 (p. 38)。

对于包含使用 EC2 启动类型的的任务或服务的集群，您的容器实例必须运行版本 1.29.0 或更高版本的 Amazon ECS 代理。有关更多信息，请参阅 Amazon ECS 容器代理版本 (p. 207)。

使用控制台为您账户上的所有 IAM 用户和角色选择已启用 Container Insights 的集群

1. 以账户的根用户身份打开位于 https://console.aws.amazon.com/ecs/ 的 Amazon ECS 控制台。
2. 在屏幕顶部的导航栏中，选择要为其选定已启用 Container Insights 的集群的区域。
3. 从控制面板中选择账户设置。
4. 对于 IAM 用户或角色，确保选择了您的根用户或容器实例 IAM 角色。
5. 对于 Container Insights，选中对应复选框。完成后，选择保存。

   > **Important**
   >
   > IAM 用户和 IAM 角色需要 `ecs:PutAccountSetting` 权限来执行此操作。

6. 在确认屏幕上，选择 Confirm (确认) 以保存选择。

使用命令行为您账户上的所有 IAM 用户和角色选择已启用 Container Insights 的集群

账户上的任意用户可以使用下列命令之一修改账户中所有 IAM 用户或角色的默认账户设置。这些更改将应用于整个 AWS 账户，除非一个 IAM 用户或角色显式覆盖自己的这些设置。

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name containerInsights --value enabled --region us-
east-1
```

- Write-ECSAccountSettingDefault (适用于 Windows PowerShell 的 AWS 工具)

```
Write-ECSAccountSettingDefault -Name containerInsights -Value enabled -Region us-east-1 -
Force
```

使用命令行以根用户身份为 IAM 用户或容器实例 IAM 角色选择已启用 Container Insights 的集群

账户的根用户可以使用以下命令之一，并在请求中指定委托人 IAM 用户或容器实例 IAM 角色的 ARN 来修改账户设置。

- put-account-setting (AWS CLI)

  以下示例用于修改特定 IAM 用户的账户设置：

```
aws ecs put-account-setting --name containerInsights --value enabled --principal-arn
 arn:aws:iam::aws_account_id:user/userName --region us-east-1
```

- Write-ECSAccountSetting (适用于 Windows PowerShell 的 AWS 工具)

以下示例用于修改特定 IAM 用户的账户设置：

```
Write-ECSAccountSetting -Name containerInsights -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/userName -Region us-east-1 -Force
```

使用 Amazon ECS 记录 AWS CloudTrail API 调用

Amazon ECS 与 AWS CloudTrail 集成，后者是一项在 Amazon ECS 中提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 对 Amazon ECS 的所有 API 调用（包括来自 Amazon ECS 控制台的调用和对 Amazon ECS API 操作的代码调用）均作为事件捕获。

如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 Amazon ECS 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台的 Event history (事件历史记录) 中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 Amazon ECS 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

有关更多信息，请参阅 AWS CloudTrail User Guide。

CloudTrail 中的 Amazon ECS 信息

在您创建 CloudTrail 账户时，即针对该账户启用了 AWS。Amazon ECS 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 Amazon ECS 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪在 AWS 分区中记录来自所有区域的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个区域的 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Amazon ECS 操作均由 CloudTrail 记录下来并记载到 Amazon Elastic Container Service API Reference 中。例如，对 `CreateService`、 `RunTask` 和 `DeleteCluster` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Amazon ECS 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、

请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

> **Note**
>
> 为提高可读性，这些示例已进行格式化处理。在 CloudTrail 日志文件，所有条目和事件都连接成一行。此外，该示例限于一个 Amazon ECS 条目。在实际的 CloudTrail 日志文件中，有来自多个 AWS 服务的条目和事件。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateCluster` 操作：

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-20T18:32:25Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Mary_Major"
            }
        }
    },
    "eventTime": "2018-06-20T19:04:36Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "clusterName": "default"
    },
    "responseElements": {
        "cluster": {
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
            "pendingTasksCount": 0,
            "registeredContainerInstancesCount": 0,
            "status": "ACTIVE",
            "runningTasksCount": 0,
            "statistics": [],
            "clusterName": "default",
            "activeServicesCount": 0
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Amazon Elastic Container Service 中的安全性

AWS 的云安全性的优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的 安全性和云中 的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 Amazon Elastic Container Service 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 – 您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon ECS 时应用责任共担模型。以下主题说明如何配置 Amazon ECS 以实现您的安全性和合规性目标。您还将了解如何使用其他 AWS 服务来帮助您监控和保护 Amazon ECS 资源。

主题

- 适用于 Amazon Elastic Container Service 的 Identity and Access Management (p. 367)

适用于 Amazon Elastic Container Service 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）以使用 Amazon ECS 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

- 受众 (p. 368)
- 使用身份进行身份验证 (p. 368)
- 使用策略管理访问 (p. 369)
- Amazon Elastic Container Service 如何与 IAM 协同工作 (p. 370)
- Amazon Elastic Container Service 基于身份的策略示例 (p. 374)
- Amazon ECS API 操作支持的资源级权限 (p. 385)
- 对 Amazon ECS 使用服务相关角色 (p. 392)
- 托管策略和信任关系 (p. 396)
- Amazon ECS 容器实例 IAM 角色 (p. 404)
- Amazon ECS 任务执行 IAM 角色 (p. 406)
- Amazon ECS 服务计划程序 IAM 角色 (p. 410)
- Amazon ECS CodeDeploy IAM 角色 (p. 411)
- Amazon ECS 服务 Auto Scaling IAM 角色 (p. 414)

- CloudWatch Events IAM 角色 (p. 416)
- Amazon ECS 任务角色 (p. 418)
- 任务的 IAM 角色 (p. 418)
- 故障排除 Amazon Elastic Container Service 身份和访问 (p. 422)

受众

如何使用 AWS Identity and Access Management (IAM) 因您可以在 Amazon ECS 中执行的操作而异。

服务用户 – 如果您使用 Amazon ECS 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Amazon ECS 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon ECS 中的一项功能，请参阅故障排除 Amazon Elastic Container Service 身份和访问 (p. 422)。

服务管理员 – 如果您在公司负责管理 Amazon ECS 资源，则您可能具有 Amazon ECS 的完全访问权限。您有责任确定您的员工应访问哪些 Amazon ECS 功能和资源。然后，您必须向 IAM 管理员提交请求以更改您的服务用户的权限。检查此页上的信息，了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon ECS 搭配使用的更多信息，请参阅Amazon Elastic Container Service 如何与 IAM 协同工作 (p. 370)。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解有关您可以如何编写策略以管理 Amazon ECS 访问权限的详细信息。要查看您可在 IAM 中使用的基于身份的 Amazon ECS 示例策略，请参阅Amazon Elastic Container Service 基于身份的策略示例 (p. 374)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。有关使用 AWS 管理控制台登录的更多信息，请参阅 IAM 用户指南 中的 IAM 控制台和登录页面。

您必须以 AWS 账户根用户、IAM 用户身份或通过代入 IAM 角色进行身份验证（登录到 AWS）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些案例中，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其他公司的凭证访问 AWS 时，您间接地代入了角色。

要直接登录到 AWS 管理控制台，请使用您的密码和 根用户 电子邮件或 IAM 用户名。您可以使用 根用户 或 IAM 用户访问密钥以编程方式访问 AWS。AWS 提供了开发工具包和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。使用签名版本 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅 AWS General Reference 中的签名版本 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅 IAM 用户指南 中的在 AWS 中使用多重身份验证 (MFA)。

AWS 账户根用户

当您首次创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源有完全访问权限的单点登录身份。此身份称为 AWS 账户 根用户，可使用您创建账户时所用的电子邮件地址和密码登录来获得此身份。强烈建议您不使用 根用户 执行日常任务，即使是管理任务。请遵守仅将 根用户 用于创建首个 IAM 用户的最佳实践。然后请妥善保存 根用户 凭证，仅用它们执行少数账户和服务管理任务。

IAM 用户和群组

IAM 用户是 AWS 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可以拥有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组 是指定一个 IAM 用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您有一个名为 IAMAdmins 的组并为该组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南 中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色 是 AWS 账户中具有特定权限的实体。它类似于 IAM 用户，但未与特定人员关联。您可以通过切换角色，在 AWS 管理控制台中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以代入角色。有关使用角色方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 临时 IAM 用户权限 – IAM 用户可代入 IAM 角色，暂时获得针对特定任务的不同权限。
- 联合身份用户访问 – 您也可以不创建 IAM 用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的现有身份。这些用户被称为联合身份用户。在通过身份提供商请求访问权限时，AWS 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。
- 跨账户访问 – 您可以使用 IAM 角色允许其他账户中的某个人（可信任委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。
- AWS 服务访问 –服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。在设置一些 AWS 服务环境时，您必须为服务定义要代入的角色。这个服务角色必须包含该服务访问所需的 AWS 资源会用到的所有必要权限。服务角色因服务而异，但只要您满足服务记录在案的要求，许多服务都允许您选择权限。服务角色只在您的账户内提供访问权限，不能用于为访问其他账户中的服务授权。您可以在 IAM 中创建、修改和删除服务角色。例如，您可以创建一个角色，此角色允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将该存储桶中的数据加载到 Amazon Redshift 集群中。有关更多信息，请参阅 IAM 用户指南 中的创建角色以向 AWS 服务委派权限。
- 在 Amazon EC2 上运行的应用程序–对于在 EC2 实例上运行、并发出 AWS CLI 或 AWS API 请求的应用程序，您可以使用 IAM 角色管理它们的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是否使用 IAM 角色，请参阅 IAM 用户指南 中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 IAM 身份或 AWS 资源，以便控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个实体（根用户、IAM 用户或 IAM 角色）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概述。

IAM 管理员可以使用策略来指定哪些用户有权访问 AWS 资源，以及他们可以对这些资源执行哪些操作。每个 IAM 实体（用户或角色）在一开始都没有权限。换言之，默认情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM 策略定义操作的权限，无论您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS 管理控制台、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、角色或组）的 JSON 权限策略文档。这些策略控制身份可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南 中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是可以附加到 AWS 账户中的多个用户、组和角色的独立策略。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管策略与内联策略之间进行选择 。

基于资源的策略

基于资源的策略是附加到资源 (如 Amazon S3 存储桶) 的 JSON 策略文档。服务管理员可以使用这些策略来定义指定的委托人（账户成员、用户或角色）可以对该资源以及在什么条件执行哪些操作。基于资源的策略是内联策略。没有基于托管资源的策略。

访问控制列表 (ACL)

访问控制策略 (ACL) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACL 类似于基于资源的策略，但它们是唯一不使用 JSON 策略文档格式的策略类型。Amazon S3、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅Amazon Simple Storage Service 开发人员指南中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一项高级功能，借助该功能，您可以设置基于身份的策略可以授予 IAM 实体的最大权限（IAM 用户或角色）。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南 中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 是一项服务，用于分组和集中管理您的企业拥有的多个 AWS 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 组织 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南 中的 SCP 工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南 中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多个策略类型时是否允许请求，请参阅 IAM 用户指南 中的策略评估逻辑。

Amazon Elastic Container Service 如何与 IAM 协同工作

使用 IAM 来管理 Amazon ECS 的访问权限之前，您应该了解哪些 IAM 功能可与 Amazon ECS 协同工作。要概括了解 Amazon ECS 及其他 AWS 服务如何与 IAM 协同工作，请参阅 IAM 用户指南 中的可与 IAM 协同工作的 AWS 服务。

主题

- Amazon ECS 基于身份的策略 (p. 371)
- Amazon ECS 基于资源的策略 (p. 373)
- 基于 Amazon ECS 标签的授权 (p. 373)
- Amazon ECS IAM 角色 (p. 374)

Amazon ECS 基于身份的策略

使用 IAM 基于身份的策略，您可以指定允许或拒绝操作和资源，以及指定在什么条件下允许或拒绝操作。Amazon ECS 支持特定操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 IAM JSON 策略元素参考。

操作

基于 IAM 身份的策略的 `Action` 元素描述该策略将允许或拒绝的特定操作。策略操作通常与关联的 AWS API 操作同名。此策略用于策略中以授予执行关联操作的权限。

Amazon ECS 中的策略操作在操作前使用以下前缀：`ecs:`。例如，要授予某人使用 Amazon ECS `CreateCluster` API 操作创建 Amazon ECS 集群的权限，您应将 `ecs:CreateCluster` 操作纳入其策略中。策略语句必须包括 `Action` 或 `NotAction` 元素。Amazon ECS 定义了自己的一组操作，描述了您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "ecs:action1",
      "ecs:action2"
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "ecs:Describe*"
```

要查看 Amazon ECS 操作的列表，请参阅 Amazon ECS API 操作支持的资源级权限 (p. 385)。

资源

`Resource` 元素指定要向其应用操作的对象。语句必须包含 `Resource` 或 `NotResource` 元素。您可使用 ARN 来指定资源，或使用通配符 (*) 以指明该语句适用于所有资源。

Amazon ECS 集群资源具有以下 ARN：

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

有关 ARN 格式的更多信息，请参阅 Amazon 资源名称 (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 `my-cluster` 容器，请使用以下 ARN：

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster"
```

要指定属于特定账户的所有集群，请使用通配符 (*)：

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/*"
```

某些 Amazon ECS 操作（例如用于创建资源的那些操作）不能在特定资源上执行。在这些情况下，您必须使用通配符 (*)。

```
"Resource": "*"
```

一些 Amazon ECS API 操作可以在多个资源上执行。例如，在调用 `DescribeClusters` API 操作时可以引用多个集群。要在单个语句中指定多个资源，请使用逗号分隔其 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

下表介绍了 Amazon ECS API 操作使用的每种类型资源的 ARN。

> **Important**
>
> 下表为 Amazon ECS 任务、服务和容器实例使用新的较长的 ARN 格式。如果您未选择使用长 ARN 格式，ARN 将不包含群集名称。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。

资源类型	ARN
所有 Amazon ECS 资源	arn:aws:ecs:*
指定账户在指定地区拥有的所有 Amazon ECS 资源	arn:aws:ecs:region:account:*
群集	arn:aws:ecs:region:account:cluster/cluster-name
容器实例	arn:aws:ecs:region:account:container-instance/cluster-name/container-instance-id
任务定义	arn:aws:ecs:region:account:task-definition/task-definition-family-name:task-definition-revision-number
服务	arn:aws:ecs:region:account:service/cluster-name/service-name
任务	arn:aws:ecs:region:account:task/cluster-name/task-id
容器	arn:aws:ecs:region:account:container/container-id

要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon ECS API 操作支持的资源级权限 (p. 385)。

条件键

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以构建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 `OR` 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，仅当 IAM 用户使用其 IAM 用户名进行标记时，您才可为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM 策略元素：变量和标签。

Amazon ECS 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南 中的 AWS 全局条件上下文键。

Amazon ECS 实施以下特定于服务的条件键。

条件键	描述	评估类型
aws:RequestTag/ ${TagKey}	上下文键的格式为 `"aws:RequestTag/tag-key":"tag-value"`，其中 *tag-key* 和 *tag-value* 是标签键值对。 检查 AWS 请求中是否存在目标键值对。例如，您可以检查请求是否包含标签键 `"Dept"` 并具有 `"Accounting"` 值。	字符串
aws:ResourceTag/ ${TagKey}	上下文键的格式为 `"aws:ResourceTag/tag-key":"tag-value"`，其中 *tag-key* 和 *tag-value* 是标签键值对。 检查附加到身份资源（用户或角色）的标签是否与指定的键名称和键值匹配。	字符串
aws:TagKeys	该上下文键的格式为 `"aws:TagKeys":"tag-key"`，其中 *tag-key* 是没有值的标签键列表 (例如，`["Dept","Cost-Center"]`)。 检查在 AWS 请求中包含的标签键。	字符串
ecs:ResourceTag/ ${TagKey}	上下文键的格式为 `"ecs:ResourceTag/tag-key":"tag-value"`，其中 *tag-key* 和 *tag-value* 是标签键值对。 检查附加到身份资源（用户或角色）的标签是否与指定的键名称和键值匹配。	字符串
ecs:cluster	上下文键的格式为 `"ecs:cluster":"cluster-arn"`，其中 *cluster-arn* 是 Amazon ECS 集群的 ARN。	ARN，Null
ecs:container-instances	上下文键的格式为 `"ecs:container-instances":"container-instance-arns"`，其中 *container-instance-arns* 是一个或多个容器实例 ARN。	ARN，Null
ecs:task-definition	上下文键的格式为 `"ecs:task-definition":"task-definition-arn"`，其中 *task-definition-arn* 是 Amazon ECS 任务定义的 ARN。	ARN，Null
ecs:service	上下文键的格式为 `"ecs:service":"service-arn"`，其中 *service-arn* 是 Amazon ECS 服务的 ARN。	ARN，Null

要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon ECS API 操作支持的资源级权限 (p. 385)。

示例

要查看 Amazon ECS 基于身份的策略的示例，请参阅Amazon Elastic Container Service 基于身份的策略示例 (p. 374)。

Amazon ECS 基于资源的策略

Amazon ECS 不支持基于资源的策略。

基于 Amazon ECS 标签的授权

您可以将标签附加到 Amazon ECS 资源或将请求中的标签传递到 Amazon ECS。要基于标签控制访问，您需要使用 、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的 condition 元素中提供标签信息。有关更多信息，请参阅 IAM 用户指南 中的使用标签控制访问。

有关标记 Amazon ECS 资源的更多信息，请参阅资源和标签 (p. 330)。

若要查看基于身份的策略示例（基于资源上的标签来限制对该资源的访问），请参阅基于标签描述 Amazon ECS 服务 (p. 385)。

Amazon ECS IAM 角色

IAM 角色是 AWS 账户中具有特定权限的实体。

使用 Amazon ECS 的临时凭证

您可以使用临时凭证进行联合身份登录，代入 IAM 角色或代入跨账户角色。您可以通过调用 AWS STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

Amazon ECS 支持使用临时凭证。

服务相关角色

服务相关角色允许 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon ECS 支持服务相关角色。有关创建或管理 Amazon ECS 服务相关角色的详细信息，请参阅 对 Amazon ECS 使用服务相关角色 (p. 392)。

服务角色

此功能允许服务代表您代入服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并由该账户拥有。这意味着 IAM 管理员可以更改此角色的权限。但是，这样操作可能会中断服务的功能。

Amazon ECS 支持服务角色。

Amazon Elastic Container Service 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 Amazon ECS 资源的权限。它们还无法使用 AWS 管理控制台、AWS CLI 或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，为用户和角色授予权限，以便对他们所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南 中的 JSON 选项卡上的创建策略。

主题

- 策略最佳实践 (p. 375)
- 允许用户查看他们自己的权限 (p. 375)
- Amazon ECS 首次运行向导权限 (p. 376)
- 集群示例 (p. 379)
- 容器实例示例 (p. 380)
- 任务定义示例 (p. 381)
- 运行任务示例 (p. 382)
- 启动任务示例 (p. 382)
- 列出并描述任务示例 (p. 383)

- 创建服务示例 (p. 383)
- 更新服务示例 (p. 384)
- 基于标签描述 Amazon ECS 服务 (p. 385)

策略最佳实践

基于身份的策略非常强大。它们确定某个人是否可以创建、访问或删除您账户中的 Amazon ECS 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略 – 要快速开始使用 Amazon ECS，请使用 AWS 托管策略，为您的员工授予他们所需的权限。这些策略已在您的账户中提供，并由 AWS 维护和更新。有关更多信息，请参阅 IAM 用户指南 中的利用 AWS 托管策略开始使用权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其他权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最小权限。
- 为敏感操作启用 MFA – 为增强安全性，要求 IAM 用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。有关更多信息，请参阅 IAM 用户指南 中的在 AWS 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 IAM 用户指南 中的 IAM JSON 策略元素：Condition。

允许用户查看他们自己的权限

此示例显示您可以如何创建策略，以便允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "ViewOwnUserInfo",
                "Effect": "Allow",
                "Action": [
                    "iam:GetUserPolicy",
                    "iam:ListGroupsForUser",
                    "iam:ListAttachedUserPolicies",
                    "iam:ListUserPolicies",
                    "iam:GetUser"
                ],
                "Resource": [
                    "arn:aws:iam::*:user/${aws:username}"
                ]
            },
            {
                "Sid": "NavigateInConsole",
                "Effect": "Allow",
                "Action": [
                    "iam:GetGroupPolicy",
                    "iam:GetPolicyVersion",
                    "iam:GetPolicy",
                    "iam:ListAttachedGroupPolicies",
                    "iam:ListGroupPolicies",
                    "iam:ListPolicyVersions",
                    "iam:ListPolicies",
                    "iam:ListUsers"
                ],
```

```
                "Resource": "*"
            }
        ]
    }
```

Amazon ECS 首次运行向导权限

Amazon ECS 首次运行向导简化了创建集群并运行您的任务和服务的过程。不过，用户需要通过多种 AWS 服务来获得执行许多 API 操作的权限才能完成此向导。以下 AmazonECS_FullAccess (p. 397) 托管策略说明完成 Amazon ECS 首次运行向导所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:Describe*",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "lambda:ListFunctions",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
```

```
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CancelSpotFleetRequests",
                "ec2:CreateInternetGateway",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSubnet",
                "ec2:DeleteVpc",
                "ec2:Describe*",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:RunInstances",
                "ec2:RequestSpotFleet",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "ecs:*",
                "events:DescribeRule",
                "events:DeleteRule",
                "events:ListRuleNamesByTarget",
                "events:ListTargetsByRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "iam:ListAttachedRolePolicies",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:FilterLogEvents",
                "route53:GetHostedZone",
                "route53:ListHostedZonesByName",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetHealthCheck",
                "servicediscovery:CreatePrivateDnsNamespace",
                "servicediscovery:CreateService",
                "servicediscovery:GetNamespace",
                "servicediscovery:GetOperation",
                "servicediscovery:GetService",
                "servicediscovery:ListNamespaces",
                "servicediscovery:ListServices",
                "servicediscovery:UpdateService"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ssm:GetParametersByPath",
                "ssm:GetParameters",
                "ssm:GetParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/aws/service/ecs*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteInternetGateway",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-name": "EC2ContainerService-
*"
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ecs-tasks.amazonaws.com"
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/ecsInstanceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/ecsAutoscaleRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "application-autoscaling.amazonaws.com",
                        "application-autoscaling.amazonaws.com.cn"
                    ]
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "ecs.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "ecs.application-autoscaling.amazonaws.com",
                        "autoscaling.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

首次运行向导还会尝试自动创建不同的 IAM 角色，具体取决于所用的任务启动类型。例如，Amazon ECS 服务角色、容器实例 IAM 角色和任务执行 IAM 角色。要确保首次运行时能够创建这些 IAM 角色，必须满足以下条件之一：

- 您的用户拥有管理员权限。有关更多信息，请参阅 Amazon ECS 的设置 (p. 7)。
- 您的用户拥有创建服务角色的 IAM 权限。有关更多信息，请参阅创建角色以向 AWS 服务委派权限。
- 您让一个拥有管理员权限的用户手动创建所需的 IAM 角色，使之对所用的账户可用。有关更多信息，请参阅下列内容：
  - Amazon ECS 服务计划程序 IAM 角色 (p. 410)
  - Amazon ECS 容器实例 IAM 角色 (p. 404)
  - Amazon ECS 任务执行 IAM 角色 (p. 406)

集群示例

以下 IAM 策略授予创建和列出集群所需的权限。`CreateCluster` 和 `ListClusters` 操作不接受任何资源，因此，所有资源的资源定义将设置为 `*`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

以下 IAM 策略授予描述和删除特定集群所需的权限。`DescribeClusters` 和 `DeleteCluster` 操作将集群 ARN 接受为资源。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeCluster",
        "ecs:DeleteCluster"
      ],
      "Resource": [
        "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/<cluster_name>"
      ]
    }
  ]
}
```

以下 IAM 策略可附加到用户或组，该策略仅允许用户或组对特定集群执行操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecs:Describe*",
        "ecs:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "ecs:DeleteCluster",
        "ecs:DeregisterContainerInstance",
        "ecs:ListContainerInstances",
        "ecs:RegisterContainerInstance",
        "ecs:SubmitContainerStateChange",
        "ecs:SubmitTaskStateChange"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
    },
    {
      "Action": [
        "ecs:DescribeContainerInstances",
        "ecs:DescribeTasks",
        "ecs:ListTasks",
        "ecs:UpdateContainerAgent",
        "ecs:StartTask",
        "ecs:StopTask",
        "ecs:RunTask"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
        }
      }
    }
  ]
}
```

容器实例示例

虽然容器实例注册由 Amazon ECS 代理处理，但有时您可能需要允许用户从集群中手动取消注册实例。容器实例可能已意外注册到错误的集群，或者实例已终止，但仍有任务在其上运行。

以下 IAM 策略可让用户列出并取消注册指定集群中的容器实例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeregisterContainerInstance",
        "ecs:ListContainerInstances"
      ],
      "Resource": [
        "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
      ]
    }
  ]
}
```

以下 IAM 策略可让用户描述指定集群中的指定容器实例：要对集群中的所有容器实例开放此权限，您可以将容器实例 UUID 替换为 *。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeContainerInstances"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
        }
      },
      "Resource": [
        "arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<container_instance_UUID>"
      ]
    }
  ]
}
```

任务定义示例

任务定义 IAM 策略不支持资源级权限，但以下 IAM 策略可让用户注册、列出和描述任务定义：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RegisterTaskDefinition",
        "ecs:ListTaskDefinitions",
        "ecs:DescribeTaskDefinition"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
```

```
}
```

运行任务示例

任务定义是 RunTask 的资源。要限制用户可在其上运行任务定义的集群，您可以在 Condition 数据块中指定这些集群。这样做的好处是，您不必在资源中列出任务定义和集群以允许适当的访问。您可以应用任务定义和/或集群。

以下 IAM 策略授予在特定集群上运行特定任务定义的任意修订的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
        }
      },
      "Resource": [
        "arn:aws:ecs:<region>:<aws_account_id>:task-definition/<task_family>:*"
      ]
    }
  ]
}
```

启动任务示例

任务定义是 StartTask 的资源。要限制用户可在其中启动任务定义的集群和容器实例，您可以在 Condition 数据块中指定它们。这样做的好处是，您不必在资源中列出任务定义和集群以允许适当的访问。您可以应用任务定义和/或集群。

以下 IAM 策略授予在特定集群和特定容器实例上开始对特定任务定义进行任意修订的权限。

> **Note**
>
> 在本示例中，在通过 AWS CLI 或其他 AWS 开发工具包调用 StartTask API 时，必须指定任务定义修订，以便 Resource 映射匹配。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:StartTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>",
          "ecs:container-instances" : [
            "arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<container_instance_UUID>"
          ]
        }
      },
      "Resource": [
```

```
        "arn:aws:ecs:<region>:<aws_account_id>:task-definition/<task_family>:*"
      ]
    }
  ]
}
```

列出并描述任务示例

以下 IAM 策略可让用户列出指定集群的任务：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:ListTasks"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
        }
      },
      "Resource": [
        "*"
      ]
    }
  ]
}
```

以下 IAM 策略可让用户描述指定集群中的指定任务：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeTasks"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"
        }
      },
      "Resource": [
        "arn:aws:ecs:<region>:<aws_account_id>:task/<task_UUID>"
      ]
    }
  ]
}
```

创建服务示例

以下 IAM 策略允许用户在 AWS 管理控制台中创建 Amazon ECS 服务：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:Describe*",
        "application-autoscaling:PutScalingPolicy",
        "application-autoscaling:RegisterScalableTarget",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "ecs:List*",
        "ecs:Describe*",
        "ecs:CreateService",
        "elasticloadbalancing:Describe*",
        "iam:AttachRolePolicy",
        "iam:CreateRole",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:ListAttachedRolePolicies",
        "iam:ListRoles",
        "iam:ListGroups",
        "iam:ListUsers"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

更新服务示例

以下 IAM 允许 AWS 管理控制台中更新 Amazon ECS 服务：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:Describe*",
        "application-autoscaling:PutScalingPolicy",
        "application-autoscaling:DeleteScalingPolicy",
        "application-autoscaling:RegisterScalableTarget",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "ecs:List*",
        "ecs:Describe*",
        "ecs:UpdateService",
        "iam:AttachRolePolicy",
        "iam:CreateRole",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:ListAttachedRolePolicies",
        "iam:ListRoles",
        "iam:ListGroups",
        "iam:ListUsers"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

基于标签描述 Amazon ECS 服务

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon ECS 资源的访问。此示例显示如何创建允许描述服务的策略。但是，仅当服务标签 `Owner` 具有该用户的用户名的值时，才授予此权限。此策略还授予在控制台上完成此操作的必要权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DescribeServices",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "*"
        },
        {
            "Sid": "ViewServiceIfOwner",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "arn:aws:ecs:*:*:service/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以将此策略附加到您账户中的 IAM 用户。如果名为 `richard-roe` 的用户尝试描述 Amazon ECS 服务，则服务必须标记为 `Owner=richard-roe` 或 `owner=richard-roe`。否则，他会被拒绝访问。条件标签键 `Owner` 匹配 `Owner` 和 `owner`，因为条件键名称不区分大小写。有关更多信息，请参阅 IAM 用户指南中的 IAM JSON 策略元素：Condition。

Amazon ECS API 操作支持的资源级权限

资源级权限 一词是指指定允许用户对哪些资源执行操作的能力。Amazon ECS 对资源级权限提供部分支持。这意味着对于某些 Amazon ECS 操作，您可以控制何时允许用户执行操作（基于必须满足的条件）或是允许用户使用的特定资源。例如，您可以为用户授予启动实例的权限，但是仅限特定类型的实例，并且只能使用特定的 AMI。

有关资源级权限的注意事项

当通过在 IAM 策略中指定资源的 Amazon 资源名称 (ARN) 来控制对 Amazon ECS API 操作的访问时，请记住，ECS 引入了一个账户设置，该设置会影响容器实例、服务和任务的 ARN 格式。要使用资源级权限，我们建议您选择使用新的、更长的 ARN 格式。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。

在评估 IAM 策略时，根据指定资源对新的、更长的 ARN 格式的使用来评估这些资源。以下是访问控制方式的示例。

使用通配符指定仅具有集群的服务

示例：`arn:aws:ecs:`*`region`*`:`*`aws_account_id`*`:service/`*`cluster_name`*`*`

在此示例中，将控制对以下服务的访问：

- `cluster_name*` 集群中所有使用新 ARN 格式的服务。
- `cluster_name*` 集群中所有使用旧 ARN 格式的服务。

> **Important**
>
> 这将不会控制对使用旧 ARN 格式、具有前缀为 `cluster_name` 的服务名称且不在 `cluster_name*` 集群中的服务的访问。

使用通配符指定同时具有集群和服务名称的服务

示例：`arn:aws:ecs:region:aws_account_id:service/cluster_name/service_name*`

在此示例中，将控制对以下服务的访问：

- `cluster_name` 集群中所有使用新 ARN 格式的服务（具有 `service_name` 前缀）。
- `cluster_name` 集群中所有使用旧 ARN 格式的服务（具有 `service_name` 前缀），即使服务的实际 ARN 仍将采用 `arn:aws:ecs:region:aws_account_id:service/service_name*` ARN 格式。

指定具有完整 ARN 的服务

示例：`arn:aws:ecs:region:aws_account_id:service/cluster_name/service_name`

在此示例中，将控制对以下服务的访问：

- `cluster_name` 集群中所有使用新 ARN 格式的服务（具有 `service_name` 服务名称）。
- `cluster_name` 集群中所有使用旧 ARN 格式的服务（具有 `service_name` 服务名称），即使服务的实际 ARN 仍将采用 `arn:aws:ecs:region:aws_account_id:service/service_name` ARN 格式。

资源级权限

下表介绍当前支持资源级权限的 Amazon ECS API 操作，以及每个操作支持的资源、资源 ARN 和条件密钥。

> **Important**
>
> 如果此表中的 Amazon ECS API 操作未针对 `Resource` 和 `Condition keys` 标记为 `N/A`，则它不支持资源级权限。您可以向用户授予使用该操作的权限，但应为策略语句的资源元素指定 `*`。

API 操作	资源	条件键
CreateCluster	不适用	aws:RequestTag/${TagKey} aws:TagKeys
CreateService	Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:RequestTag/${TagKey} aws:ResourceTag ecs:ResourceTag ecs:task-definition ecs:cluster aws:TagKeys
CreateTaskSet	不适用	ecs:service ecs:task-definition ecs:cluster

API 操作	资源	条件键
DeleteAccountSetting	不适用	不适用
DeleteAttributes	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag aws:TagKeys ecs:cluster
DeleteCluster	集群 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster*	aws:ResourceTag ecs:ResourceTag
DeleteService	Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:ResourceTag ecs:ResourceTag ecs:cluster
DeleteTaskSet	TaskSet arn:aws:ecs:*region*:*aws_account_id*:cluster/*service-name*/*taskset-id*	ecs:service ecs:cluster
DeregisterContainerInstance	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster*	aws:ResourceTag ecs:ResourceTag
DeregisterTaskDefinition	不适用	不适用
DescribeClusters	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster1*, arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster2*	aws:ResourceTag ecs:ResourceTag
DescribeContainerInstances	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag aws:TagKeys ecs:cluster
DescribeServices	Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:ResourceTag ecs:ResourceTag ecs:cluster
DescribeTaskDefinition	不适用	不适用
DescribeTasks	Task arn:aws:ecs:*region*:*aws_account_id*:task/*cluster-name*/*task-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster

API 操作	资源	条件键
DescribeTaskSets	TaskSet arn:aws:ecs:*region*:*aws_account_id*:clusters/*service-name*/*taskset-id*	ecs:service ecs:cluster
DiscoverPollEndpoint	不适用	不适用
ListAccountSettings	不适用	不适用
ListAttributes	群集 arn:aws:ecs:*region*:*aws_account_id*:clusters/*my-cluster*	aws:ResourceTag ecs:ResourceTag
ListClusters	不适用	不适用
ListContainerInstances	群集 arn:aws:ecs:*region*:*aws_account_id*:clusters/*my-cluster*	aws:ResourceTag ecs:ResourceTag
ListServices	不适用	ecs:cluster
ListTagsForResource	群集 arn:aws:ecs:*region*:*aws_account_id*:clusters/*my-cluster* Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id* Task arn:aws:ecs:*region*:*aws_account_id*:task/*cluster-name*/*task-id* 任务定义 arn:aws:ecs:*region*:*aws_account_id*:task-definition/*hello_world:8*	aws:ResourceTag ecs:ResourceTag
ListTaskDefinitionFamilies	不适用	不适用
ListTaskDefinitions	不适用	不适用
ListTasks	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster
轮询	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster

API 操作	资源	条件键
PutAccountSetting	不适用	不适用
PutAccountSettingDefault	不适用	不适用
PutAttributes	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag aws:ResourceTag ecs:cluster
RegisterContainerInstance	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster*	aws:RequestTag/${TagKey} aws:ResourceTag ecs:ResourceTag aws:TagKeys
RegisterTaskDefinition	不适用	aws:RequestTag/${TagKey} aws:TagKeys
RunTask	任务定义 arn:aws:ecs:*region*:*aws_account_id*:task-definition/*hello_world:8*	aws:RequestTag/${TagKey} aws:ResourceTag ecs:ResourceTag aws:TagKeys ecs:cluster
StartTask	任务定义 arn:aws:ecs:*region*:*aws_account_id*:task-definition/*hello_world:8*	aws:RequestTag/${TagKey} aws:ResourceTag ecs:ResourceTag aws:TagKeys ecs:cluster ecs:container-instances
StartTelemetrySession	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag aws:ResourceTag ecs:cluster
StopTask	Task arn:aws:ecs:*region*:*aws_account_id*:task/*cluster-name*/*task-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster

API 操作	资源	条件键
SubmitContainerStateChange	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster*	aws:ResourceTag ecs:ResourceTag
SubmitTaskStateChange	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster*	aws:ResourceTag ecs:ResourceTag
TagResource	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster* Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id* Task arn:aws:ecs:*region*:*aws_account_id*:task/*cluster-name*/*task-id* 任务定义 arn:aws:ecs:*region*:*aws_account_id*:task-definition/*hello_world:8* Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:RequestTag/${TagKey} aws:ResourceTag ecs:ResourceTag aws:TagKeys

API 操作	资源	条件键
UntagResource	群集 arn:aws:ecs:*region*:*aws_account_id*:cluster/*my-cluster* Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id* Task arn:aws:ecs:*region*:*aws_account_id*:task/*cluster-name*/*task-id* 任务定义 arn:aws:ecs:*region*:*aws_account_id*:task-definition/*hello_world:8* Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:ResourceTag ecs:ResourceTag aws:TagKeys
UpdateContainerAgent	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster
UpdateContainerInstancesState	Container instance arn:aws:ecs:*region*:*aws_account_id*:container-instance/*cluster-name*/*container-instance-id*	aws:ResourceTag ecs:ResourceTag ecs:cluster
UpdateService	Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:ResourceTag ecs:ResourceTag ecs:cluster ecs:task-definition
UpdateServicePrimaryTaskSet	Service arn:aws:ecs:*region*:*aws_account_id*:service/*cluster-name*/*service-name*	aws:ResourceTag ecs:ResourceTag ecs:cluster
UpdateTaskSet	TaskSet arn:aws:ecs:*region*:*aws_account_id*:cluster/*service-name*/*taskset-id*	ecs:cluster ecs:service

对 Amazon ECS 使用服务相关角色

Amazon Elastic Container Service 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与 Amazon ECS 直接关联的独特类型的 IAM 角色。服务相关角色由 Amazon ECS 预定义，并包含该服务代表您调用其他 AWS 服务所需的一切权限。

服务相关角色使 Amazon ECS 的设置更轻松，因为您不必手动添加必要的权限。Amazon ECS 定义其服务相关角色的权限，除非另行定义，否则仅 Amazon ECS 可以代入其角色。定义的权限包括信任策略和权限策略，并且权限策略不能附加到任何其他 IAM 实体。

只有在首先删除角色的相关资源后，才能删除角色。这将保护您的 Amazon ECS 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务并查找 Service-Linked Role 列为 Yes 的服务。选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon ECS 的服务相关角色权限

Amazon ECS 使用名为 AWSServiceRoleForECS – Role to enable Amazon ECS to manage your cluster. 的服务相关角色。

AWSServiceRoleForECS 服务相关角色信任以下服务代入该角色：

- `ecs.amazonaws.com`

角色权限策略允许 Amazon ECS 对指定资源完成以下操作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECSTaskManagement",
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets",
                "route53:ChangeResourceRecordSets",
                "route53:CreateHealthCheck",
                "route53:DeleteHealthCheck",
                "route53:Get*",
                "route53:List*",
                "route53:UpdateHealthCheck",
                "servicediscovery:DeregisterInstance",
                "servicediscovery:Get*",
                "servicediscovery:List*",
                "servicediscovery:RegisterInstance",
                "servicediscovery:UpdateInstanceCustomHealthStatus"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "ECSTagging",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        }
    ]
}
```

您必须配置权限以允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。

允许 IAM 实体创建 AWSServiceRoleForECS 服务相关角色

将以下语句添加到需要创建服务相关角色的 IAM 实体的权限策略：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ecs.amazonaws.com/
AWSServiceRoleForECS*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ecs.amazonaws.com"}}
}
```

允许 IAM 实体编辑 AWSServiceRoleForECS 服务相关角色的描述

将以下语句添加到需要编辑服务相关角色的描述的 IAM 实体的权限策略：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ecs.amazonaws.com/
AWSServiceRoleForECS*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ecs.amazonaws.com"}}
}
```

允许 IAM 实体删除 AWSServiceRoleForECS 服务相关角色

将以下语句添加到需要删除服务相关角色的 IAM 实体的权限策略：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ecs.amazonaws.com/
AWSServiceRoleForECS*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ecs.amazonaws.com"}}
}
```

为 Amazon ECS 创建服务相关角色

在大多数情况下，无需手动创建服务相关角色。例如，创建新的集群（例如，通过 Amazon ECS 首次运行、使用集群创建向导或者使用 AWS CLI 或开发工具包）时，或者在 AWS 管理控制台中创建或更新服务时，如果尚不存在服务相关角色，Amazon ECS 会为您创建。

> **Important**
>
> 创建集群的 IAM 实体必须具有相应的 IAM 权限，才能创建服务相关角色并对角色应用策略。否则，自动创建操作将失败。

在 IAM 中创建服务相关角色 (AWS CLI)

您可以使用 AWS Command Line Interface 中的 IAM 命令，借助服务代入角色时所需的信任策略和内联策略创建服务相关角色。

创建服务相关角色 (CLI)

使用以下命令：

```
$ aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

编辑 Amazon ECS 的服务相关角色

Amazon ECS 不允许您编辑 AWSServiceRoleForECS 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以编辑角色的说明。有关更多信息，请参阅 IAM 用户指南 中的修改角色。

删除 Amazon ECS 的服务相关角色

如果您不再使用 Amazon ECS，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先删除所有区域中的所有 Amazon ECS 集群才能删除服务相关角色。

清除服务相关角色

您必须先确认该角色没有活动会话并删除所有 AWS 区域中的所有 Amazon ECS 集群，然后才可以使用 IAM 删除服务相关角色。

检查服务相关角色是否有活动会话

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选择 AWSServiceRoleForECS 名称（不是复选框）。
3. 在 Summary 页面上，选择 Access Advisor，查看服务相关角色的近期活动。

   > **Note**
   >
   > 如果您不确定 Amazon ECS 是否正在使用 AWSServiceRoleForECS 角色，可以尝试删除该角色。如果服务正在使用该角色，则删除操作会失败，并且您可以查看正在使用该角色的区域。如果该角色已被使用，则您必须等待会话结束，然后才能删除该角色。您无法撤销服务相关角色对会话的权限。

删除 AWSServiceRoleForECS 服务相关角色所使用的 Amazon ECS 资源

您必须先删除所有 AWS 区域中的所有 Amazon ECS 集群，才能删除 AWSServiceRoleForECS 角色。

1. 将所有区域中的所有 Amazon ECS 服务缩小到预期数量 0，然后删除该服务。有关更多信息，请参阅 更新服务 (p. 326) 和 删除服务 (p. 328)。
2. 强制从所有区域的所有集群中取消注册所有容器实例。有关更多信息，请参阅 取消注册容器实例 (p. 196)。
3. 删除所有区域的所有 Amazon ECS 集群。有关更多信息，请参阅删除集群 (p. 41)。

在 IAM 中删除服务相关角色 (控制台)

您可以使用 IAM 控制台删除服务相关角色。

删除服务相关角色 (控制台)

1. 登录 AWS 管理控制台 并通过以下网址打开 IAM 控制台 https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择 Roles。然后选中 AWSServiceRoleForECS 旁边的复选框，而不是选择名称或行本身。
3. 选择删除角色。
4. 在确认对话框中，查看上次访问服务数据，该数据显示每个选定角色上次访问 AWS 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续，请选择 Yes, Delete 以提交服务相关角色进行删除。
5. 观察 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色的删除是异步操作，因此，在提交要删除的角色后，删除任务可能会成功，也可能会失败。

   - 如果任务成功，则角色将从列表中删除，并会在页面顶部显示成功通知。
   - 如果任务失败，您可以从通知中选择 View details 或 View Resources 以了解删除失败的原因。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除操作。

     Note

     您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。
   - 如果任务失败，并且通知不包含资源列表，则服务可能不会返回该信息。要了解如何清除该服务的资源，请参阅与 IAM 配合使用的 AWS 服务。在表中查找您的服务，然后选择是链接以查看该服务的服务相关角色文档。

在 IAM 中删除服务相关角色 (AWS CLI)

您可以从 AWS Command Line Interface 使用 IAM 命令删除服务相关角色。

删除服务相关角色 (CLI)

1. 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `deletion-task-id` 以检查删除任务的状态。输入以下命令以提交服务相关角色的删除请求：

   ```
   $ aws iam delete-service-linked-role --role-name AWSServiceRoleForECS+OPTIONAL-SUFFIX
   ```

2. 使用以下命令以检查删除任务的状态：

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除操作。

   Note

   您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资

源、其中一些资源，也可能不报告任何资源。要了解如何为不报告任何资源的服务清除资源，请参阅与 IAM 配合使用的 AWS 服务。在表中查找您的服务，然后选择是链接以查看该服务的服务相关角色文档。

在 IAM 中删除服务相关角色 (AWS API)

您可以使用 IAM API 删除服务相关角色。

删除服务相关角色 (API)

1. 要提交服务相关角色的删除请求，请调用 DeleteServiceLinkedRole。在请求中，指定 AWSServiceRoleForECS 角色名称。

   如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。
2. 要检查删除的状态，请调用 GetServiceLinkedRoleDeletionStatus。在请求中，指定 `DeletionTaskId`。

   删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除。

   Note

   您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。要了解如何为不报告任何资源的服务清除资源，请参阅 与 IAM 配合使用的 AWS 服务。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

托管策略和信任关系

Amazon ECS 和 Amazon ECR 提供了一些托管策略和信任关系，您可以将它们附加到 IAM 用户、EC2 实例和 Amazon ECS 任务，以实现对资源和 API 操作的不同级别的控制。您可以直接应用这些策略，或者也可以使用它们作为自行创建策略的起点。

主题

- Amazon ECS 托管策略和信任关系 (p. 396)
- Amazon ECR 托管策略 (p. 402)

Amazon ECS 托管策略和信任关系

Amazon ECS 提供了一些托管策略和信任关系，您可以将它们附加到 IAM 用户、EC2 实例或 Amazon ECS 任务，以实现对 Amazon ECS 资源和 API 操作的不同级别的控制。您可以直接应用这些策略，或者也可以使用它们作为自行创建策略的起点。有关这些策略中提到的每个 API 操作的更多信息，请参阅 Amazon Elastic Container Service API Reference 中的操作。

主题

- AmazonECS_FullAccess (p. 397)
- AmazonEC2ContainerServiceFullAccess (p. 400)
- AmazonEC2ContainerServiceforEC2Role (p. 400)
- AmazonEC2ContainerServiceRole (p. 401)

- AmazonEC2ContainerServiceAutoscaleRole (p. 401)
- AmazonEC2ContainerServiceTaskRole (p. 402)
- AmazonEC2ContainerServiceEventsRole (p. 402)

AmazonECS_FullAccess

此托管策略提供对 Amazon ECS 资源的管理访问权限，并通过对其他 AWS 服务资源的访问权限启用 ECS 功能，包括 VPC、Auto Scaling 组和 AWS CloudFormation 堆栈。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:Describe*",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "lambda:ListFunctions",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:AuthorizeSecurityGroupIngress",
```

```
                "ec2:CancelSpotFleetRequests",
                "ec2:CreateInternetGateway",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSubnet",
                "ec2:CreateVpc",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSubnet",
                "ec2:DeleteVpc",
                "ec2:Describe*",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:RunInstances",
                "ec2:RequestSpotFleet",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "ecs:*",
                "events:DescribeRule",
                "events:DeleteRule",
                "events:ListRuleNamesByTarget",
                "events:ListTargetsByRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "iam:ListAttachedRolePolicies",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:FilterLogEvents",
                "route53:GetHostedZone",
                "route53:ListHostedZonesByName",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetHealthCheck",
                "servicediscovery:CreatePrivateDnsNamespace",
                "servicediscovery:CreateService",
                "servicediscovery:GetNamespace",
                "servicediscovery:GetOperation",
                "servicediscovery:GetService",
                "servicediscovery:ListNamespaces",
                "servicediscovery:ListServices",
                "servicediscovery:UpdateService"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParametersByPath",
```

```
                "ssm:GetParameters",
                "ssm:GetParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/aws/service/ecs*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteInternetGateway",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSecurityGroup"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-name": "EC2ContainerService-
*"
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ecs-tasks.amazonaws.com"
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/ecsInstanceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/ecsAutoscaleRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "application-autoscaling.amazonaws.com",
                        "application-autoscaling.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": [
                        "ecs.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "ecs.application-autoscaling.amazonaws.com",
                        "autoscaling.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AmazonEC2ContainerServiceFullAccess

此托管策略授予对 Amazon ECS 的完全管理员访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:Describe*",
        "autoscaling:UpdateAutoScalingGroup",
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStack*",
        "cloudformation:UpdateStack",
        "cloudwatch:GetMetricStatistics",
        "ec2:Describe*",
        "elasticloadbalancing:*",
        "ecs:*",
        "events:DescribeRule",
        "events:DeleteRule",
        "events:ListRuleNamesByTarget",
        "events:ListTargetsByRule",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "iam:ListInstanceProfiles",
        "iam:ListRoles",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonEC2ContainerServiceforEC2Role

此托管策略允许 Amazon ECS 容器实例代表您调用 AWS。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:DeregisterContainerInstance",
        "ecs:DiscoverPollEndpoint",
        "ecs:Poll",
        "ecs:RegisterContainerInstance",
        "ecs:StartTelemetrySession",
        "ecs:Submit*",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonEC2ContainerServiceRole

此托管策略允许 Elastic Load Balancing 负载均衡器代表您注册和取消注册 Amazon ECS 容器实例。有关更多信息，请参阅Amazon ECS 服务计划程序 IAM 角色 (p. 410)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:Describe*",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DeregisterTargets",
        "elasticloadbalancing:Describe*",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:RegisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonEC2ContainerServiceAutoscaleRole

此托管策略允许 Application Auto Scaling 代表您增加和减少您的 Amazon ECS 服务的预期数量以响应 CloudWatch 警报。有关更多信息，请参阅Amazon ECS 服务 Auto Scaling IAM 角色 (p. 414)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1456535218000",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeServices",
                "ecs:UpdateService"
            ],
            "Resource": [
                "*"
```

```
            ]
        },
        {
            "Sid": "Stmt1456535243000",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AmazonEC2ContainerServiceTaskRole

此 IAM 信任关系策略允许您的 Amazon ECS 任务中的容器代表您调用 AWS API。有关更多信息，请参阅Amazon ECS 任务角色 (p. 418)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AmazonEC2ContainerServiceEventsRole

该策略允许 CloudWatch Events 代表您运行任务。有关更多信息，请参阅 排定的任务 (cron) (p. 263)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Amazon ECR 托管策略

Amazon ECR 提供了一些托管策略，您可以将它们附加到 IAM 用户或 EC2 实例，以实现对 Amazon ECR 资源和 API 操作的不同级别的控制。您可以直接应用这些策略，或者也可以使用它们作为自行创建策略的起点。有关这些策略中提到的每个 API 操作的更多信息，请参阅 Amazon Elastic Container Registry API Reference 中的操作。

主题

- AmazonEC2ContainerRegistryFullAccess (p. 403)
- AmazonEC2ContainerRegistryPowerUser (p. 403)
- AmazonEC2ContainerRegistryReadOnly (p. 403)

AmazonEC2ContainerRegistryFullAccess

此托管策略授予对 Amazon ECR 的完全管理员访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:*"
      ],
      "Resource": "*"
    }
  ]
}
```

AmazonEC2ContainerRegistryPowerUser

此托管策略授予对 Amazon ECR 的高级用户访问权限，从而允许对存储库进行读写访问，但不允许用户删除存储库或更改应用于存储库的策略文档。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "ecr:GetAuthorizationToken",
   "ecr:BatchCheckLayerAvailability",
   "ecr:GetDownloadUrlForLayer",
   "ecr:GetRepositoryPolicy",
   "ecr:DescribeRepositories",
   "ecr:ListImages",
   "ecr:DescribeImages",
   "ecr:BatchGetImage",
   "ecr:InitiateLayerUpload",
   "ecr:UploadLayerPart",
   "ecr:CompleteLayerUpload",
   "ecr:PutImage"
  ],
  "Resource": "*"
 }]
}
```

AmazonEC2ContainerRegistryReadOnly

此托管策略授予对 Amazon ECR 的只读访问权限，例如，能够列出存储库和存储库中的映像，还能通过 Docker CLI 从 Amazon ECR 中拉取映像。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "ecr:GetAuthorizationToken",
   "ecr:BatchCheckLayerAvailability",
```

```
    "ecr:GetDownloadUrlForLayer",
    "ecr:GetRepositoryPolicy",
    "ecr:DescribeRepositories",
    "ecr:ListImages",
    "ecr:DescribeImages",
    "ecr:BatchGetImage"
   ],
   "Resource": "*"
  }]
}
```

Amazon ECS 容器实例 IAM 角色

Amazon ECS 容器代理代表您调用 Amazon ECS API。运行该代理的容器实例需要一个 IAM 策略和角色，以便该服务了解该代理属于您。您必须先为容器实例创建它们启动时要使用的 IAM 角色，然后才能启动这些容器实例并将它们注册到集群。此要求适用于通过由 Amazon 提供的Amazon ECS-optimized AMI 启动的容器实例，也适用于您打算在其上运行代理的任何其他实例。

> **Important**
>
> 此角色仅在您使用 EC2 启动类型时适用。
>
> **Important**
>
> 在您的容器实例上运行的容器有权获得通过实例元数据提供给容器实例角色的所有权限。建议您将容器实例角色中的权限限制在下面显示的托管 `AmazonEC2ContainerServiceforEC2Role` 策略中提供的最低权限列表的范围内。如果您的任务中的容器需要此处未列出的其他权限，建议您为这些任务提供其自己的 IAM 角色。有关更多信息，请参阅任务的 IAM 角色 (p. 418)。
> 您可以通过在容器实例上运行以下 iptables 命令来阻止 `docker0` 桥上的容器获得提供给容器实例角色的权限（同时仍允许任务的 IAM 角色 (p. 418)提供的权限）；不过，在此规则生效的情况下，容器将无法查询实例元数据。请注意，此命令采用默认 Docker 桥配置，它不适用于使用 `host` 网络模式的容器。有关更多信息，请参阅网络模式 (p. 52)。
>
> ```
> sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
> ```
>
> 您必须将此 iptables 规则保存到容器实例上，使其在重启后仍然可用。对于Amazon ECS-optimized AMI，请使用以下命令。对于其他操作系统，请参阅该操作系统的相关文档。
>
> - 对于Amazon ECS-optimized Amazon Linux 2 AMI：
>
> ```
> sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
> ```
>
> - 对于Amazon ECS-optimized Amazon Linux AMI：
>
> ```
> sudo service iptables save
> ```

`AmazonEC2ContainerServiceforEC2Role` 策略如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:DeregisterContainerInstance",
        "ecs:DiscoverPollEndpoint",
```

```
        "ecs:Poll",
        "ecs:RegisterContainerInstance",
        "ecs:StartTelemetrySession",
        "ecs:Submit*",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 上述策略中的 `ecs:CreateCluster` 行是可选的，前提是要将容器实例注册到的群集已存在。如果群集不存在，则在启动容器实例之前，代理必须有权创建群集或者您能够使用 create-cluster 命令创建群集。
> 如果您省略了 `ecs:CreateCluster` 行，Amazon ECS 容器代理将无法创建群集，包括默认群集。

上述策略中的 `ecs:Poll` 行用于向代理授予连接 Amazon ECS 服务以报告状态和获取命令的权限。

在控制台首次运行体验中将自动为您创建 Amazon ECS 实例角色。但是，您应手动为容器实例附加托管 IAM 策略，以便 Amazon ECS 能够在引入未来功能和增强功能时添加这些功能的权限。使用以下过程检查并确定您的账户是否已拥有 Amazon ECS 实例角色并且附加托管 IAM 策略（如果需要）。

在 IAM 控制台中检查 `ecsInstanceRole`

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsInstanceRole`。如果该角色不存在，请使用以下过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions 选项卡。
5. 在 Managed Policies 部分，确保将 AmazonEC2ContainerServiceforEC2Role 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS 实例角色。否则，请执行以下子步骤来附加策略。
   a. 选择 Attach Policy。
   b. 在 Filter 框中，键入 AmazonEC2ContainerServiceforEC2Role 以缩小要附加的可用策略的范围。
   c. 选中 AmazonEC2ContainerServiceforEC2Role 策略左侧的框并选择 Attach Policy。
6. 选择信任关系 选项卡，然后选择编辑信任关系。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

为容器实例创建 **ecsInstanceRole** IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 AWS service (AWS 服务) 角色类型，然后选择 Elastic Container Service (弹性容器服务)。
4. 选择 Elastic Container Service 的 EC2 角色使用案例，然后选择 下一步：权限。
5. 在附加的权限策略部分，选择 AmazonEC2ContainerServiceforEC2Role，然后选择下一步：审核。
6. 对于 Role name，键入 `ecsInstanceRole`，或者也可以输入描述。
7. 检查您的角色信息，然后选择 Create role 以完成操作。

向容器实例角色添加 Amazon S3 只读访问权限

将配置信息存储在 Amazon S3 中的私有存储桶中并向您的容器实例 IAM 角色授予只读访问权限，这是一个允许在启动时配置容器实例的安全方便的方法。您可以将 `ecs.config` 文件的副本存储在私有存储桶中，使用 Amazon EC2 用户数据安装 AWS CLI，然后在实例启动时将配置信息复制到 `/etc/ecs/ecs.config`。

有关创建 `ecs.config` 文件，将该文件存储在 Amazon S3 中并使用此配置启动实例的更多信息，请参阅将容器实例配置存储在 Amazon S3 中 (p. 227)。

为容器实例角色授予 Amazon S3 只读访问权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 选择用于容器实例的 IAM 角色（此角色可能名为 `ecsInstanceRole`）。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。
4. 选择 Permissions 选项卡，然后选择 Attach policy。
5. 在 Attach policy 页面上，键入 `S3` into the Filter: Policy type 字段以缩小策略结果的范围。
6. 选中 AmazonS3ReadOnlyAccess 策略左侧的框并单击 Attach policy。

   Note

   此策略允许对所有 Amazon S3 资源进行只读访问。有关更严格的存储桶策略示例，请参阅 Amazon Simple Storage Service 开发人员指南中的存储桶策略示例。

Amazon ECS 任务执行 IAM 角色

Amazon ECS 容器代理将代表您调用 Amazon ECS API，因此，容器实例需要服务的 IAM 策略和角色，以便知道该代理属于您。此 IAM 角色称为任务执行 IAM 角色。您可以将多个任务执行角色用于与您的账户关联的不同目的。

以下是任务执行 IAM 角色的常见使用案例：

- 您的任务使用 Fargate 启动类型并且...
  - 从 Amazon ECR 推送容器映像。
  - 使用 `awslogs` 日志驱动程序。
- 您的任务使用 Fargate 或 EC2 启动类型并且...
  - 使用私有注册表身份验证。有关更多信息，请参阅私有注册表身份验证所需的 IAM 权限 (p. 408)。
  - 任务定义使用 Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数引用敏感数据。有关更多信息，请参阅 Amazon ECS 密钥所需的 IAM 权限 (p. 408)。

> **Note**
>
> 该任务执行角色由 Amazon ECS 容器代理版本 1.16.0 和更高版本支持。

Amazon ECS 提供了以下托管 `AmazonECSTaskExecutionRolePolicy` 策略，该策略包含上述常见使用案例所需的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

在 Amazon ECS 控制台首次运行体验中，将自动为您创建 Amazon ECS 任务执行角色；但是，您应手动为任务附加托管 IAM 策略，以便 Amazon ECS 能够在引入将来功能和增强功能时添加这些功能的权限。您可以使用以下过程检查并确定您的账户是否已拥有 Amazon ECS 任务执行角色并且已附加托管 IAM 策略（如果需要）。

在 IAM 控制台中检查 `ecsTaskExecutionRole`

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsTaskExecutionRole`。如果该角色不存在，请使用以下过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions。确保将 AmazonECSTaskExecutionRolePolicy 托管策略附加到该角色。如果附加该策略，则将正确配置 Amazon ECS 任务执行角色。否则，请执行以下子步骤来附加策略。
   a. 选择 Attach policy。
   b. 要缩小要附加的可用策略的范围，请为筛选条件键入 AmazonECSTaskExecutionRolePolicy。
   c. 选中 AmazonECSTaskExecutionRolePolicy 策略左侧的框并选择 Attach policy。
5. 选择信任关系，然后选择编辑信任关系。
6. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建 **`ecsTaskExecutionRole`** IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create role。
3. 在选择受信任实体的类型部分，选择 Elastic Container Service。
4. 对于 Select your use case (选择使用案例)，选择 Elastic Container Service Task (弹性容器服务任务)，然后选择 Next: Permissions (下一步: 权限)。
5. 在附加权限策略部分中，搜索 AmazonECSTaskExecutionRolePolicy，选择该策略，然后选择下一步: 审核。
6. 对于 Role Name (角色名称)，键入 `ecsTaskExecutionRole`，然后选择 Create role (创建角色)。

私有注册表身份验证所需的 IAM 权限

需要 Amazon ECS 任务执行角色才能使用私有注册表身份验证功能。这允许容器代理拉取容器映像。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。

要提供对您创建的密钥的访问权限，请将以下权限作为内联策略手动添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM 策略。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` — 仅当密钥使用自定义 KMS 密钥而不是默认密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

添加权限的示例内联策略如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
        "arn:aws:kms:<region>:<aws_account_id>:key:key_id"
      ]
    }
  ]
}
```

Amazon ECS 密钥所需的 IAM 权限

要使用 Amazon ECS 密钥功能，您必须具有 Amazon ECS 任务执行角色并在任务定义中引用它。这允许容器代理拉取必要的 AWS Systems Manager 或 Secrets Manager 资源。有关更多信息，请参阅指定敏感数据 (p. 107)。

要提供对您创建的 AWS Systems Manager Parameter Store 参数的访问权限，请将以下权限作为内联策略手动添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM 策略。

- `ssm:GetParameters` — 当您在任务定义中引用 Systems Manager Parameter Store 参数时是必需的。
- `secretsmanager:GetSecretValue` — 当您直接引用 Secrets Manager 密钥或者您的 Systems Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必需的。

- `kms:Decrypt` — 仅当您的密钥使用自定义 KMS 密钥而不是默认密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

以下示例内联策略添加所需权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/parameter_name",
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
        "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
      ]
    }
  ]
}
```

通过接口终端节点拉取 Amazon ECR 映像的 Fargate 任务的可选 IAM 权限

在 Amazon ECR 配置为使用接口 VPC 终端节点的情况下，当启动使用 Fargate l启动类型的任务（该任务从 Amazon ECR 中提取映像）时，可以限制任务对特定 VPC 或 VPC 终端节点的访问。可通过为要使用 IAM 条件键的任务创建任务执行角色来做到这一点。

使用以下 IAM 全局条件键来限制对特定 VPC 或 VPC 终端节点的访问。有关更多信息，请参阅 AWS 全局条件上下文键。

- `aws:SourceVpc`—限制对特定 VPC 的访问。
- `aws:SourceVpce`—限制对特定 VPC 终端节点的访问。

以下任务执行角色策略提供了一个添加条件键的示例：

> **Important**
>
> 无法对 `ecr:GetAuthorizationToken` API 操作应用 `aws:sourceVpc` 或 `aws:sourceVpce` 条件键，因为 GetAuthorizationToken API 调用将通过 AWS Fargate 拥有的弹性网络接口而不是任务的弹性网络接口。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": "vpce-xxxxxx",
                    "aws:sourceVpc": "vpc-xxxxx"
                }
            }
        }
    ]
}
```

Amazon ECS 服务计划程序 IAM 角色

Amazon ECS 服务计划程序将代表您调用 Amazon EC2 和 Elastic Load Balancing API，以在负载均衡器中注册和取消注册容器实例。您必须在启动服务前为其创建一个 IAM 角色，然后才能将负载均衡器附加到 Amazon ECS 服务。此要求适用于您计划用于负载均衡器的任意 Amazon ECS 服务。

大多数情况下，在控制台首次运行体验中将自动为您创建 Amazon ECS 服务角色。您可使用以下过程来检查您的账户是否已具有 Amazon ECS 服务角色。

`AmazonEC2ContainerServiceRole` 策略如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:Describe*",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DeregisterTargets",
        "elasticloadbalancing:Describe*",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:RegisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> `ec2:AuthorizeSecurityGroupIngress` 规则保留以后使用。Amazon ECS 不会自动更新与 Elastic Load Balancing 负载均衡器或 Amazon ECS 容器实例关联的安全组。

在 IAM 控制台中检查 **`ecsServiceRole`**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsServiceRole`。如果该角色不存在，请使用以下过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions 选项卡。
5. 在 Managed Policies 部分，确保将 AmazonEC2ContainerServiceRole 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS 服务角色。否则，请执行以下子步骤来附加策略。

   a. 选择 Attach Policy。
   b. 要缩小要附加的可用策略的范围，请为筛选条件键入 AmazonEC2ContainerServiceRole。
   c. 选中 AmazonEC2ContainerServiceRole 策略左侧的框并选择 Attach Policy。
6. 选择信任关系，然后选择编辑信任关系。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

为您的服务计划程序负载均衡器创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create role。
3. 在选择受信任实体的类型部分，选择 Elastic Container Service。
4. 在选择您的使用案例部分中，选择 Elastic Container Service，然后选择下一步: 权限。
5. 在附加的权限策略部分中，选择 AmazonEC2ContainerServiceRole 策略，然后选择下一步: 审核。
6. 对于角色名称，键入 `ecsServiceRole`，输入角色描述，然后选择创建角色。

Amazon ECS CodeDeploy IAM 角色

CodeDeploy 服务需要先获得代表您更新 Amazon ECS 服务的权限，然后您才能将 CodeDeploy 蓝/绿部署类型用于 Amazon ECS。这些权限是由 CodeDeploy IAM 角色 (`ecsCodeDeployRole`) 提供的。

> **Note**
>
> IAM 用户还需要使用 CodeDeploy 的权限；蓝/绿部署所需的 IAM 权限 (p. 280)中描述了这些权限。

提供了两个托管策略。以下所示的 `AWSCodeDeployRoleForECS` 策略向 CodeDeploy 授予使用相关操作更新任何资源的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "ecs:CreateTaskSet",
                "ecs:UpdateServicePrimaryTaskSet",
                "ecs:DeleteTaskSet",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
```

```
                "elasticloadbalancing:ModifyRule",
                "lambda:InvokeFunction",
                "cloudwatch:DescribeAlarms",
                "sns:Publish",
                "s3:GetObject",
                "s3:GetObjectMetadata",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

以下所示的 `AWSCodeDeployRoleForECSLimited` 策略向 CodeDeploy 授予更多有限权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "ecs:CreateTaskSet",
                "ecs:UpdateServicePrimaryTaskSet",
                "ecs:DeleteTaskSet",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:CodeDeployTopic_*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:*:*:function:CodeDeployHook_*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectMetadata",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
```

```
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

创建用于 CodeDeploy 的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create role。
3. 在 Select type of trusted entity (选择受信任实体的类型) 部分中，选择 AWS 服务。
4. 对于选择将使用此角色的服务，选择 CodeDeploy。
5. 对于选择您的使用案例，选择 CodeDeploy，然后选择下一步: 权限。
6. 选择下一步: 标签。
7. 对于添加标签(可选)，您可以向角色添加可选的 IAM 标签。完成后，选择下一步: 审核。
8. 对于角色名称，键入 `ecsCodeDeployRole`，输入可选描述，然后选择创建角色。

将所需权限添加到 Amazon ECS CodeDeploy IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在角色列表中搜索 `ecsCodeDeployRole`。如果该角色不存在，请使用上述过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
3. 在 Permissions policies (权限策略) 部分中，确保将 AWSCodeDeployRoleForECS 或 AWSCodeDeployRoleForECSLimited 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS CodeDeploy 服务角色。否则，请执行以下子步骤来附加策略。
   a. 选择附加策略。
   b. 要缩小要附加的可用策略的范围，请为筛选条件键入 AWSCodeDeployRoleForECS 或 AWSCodeDeployRoleForECSLimited。
   c. 选中 AWS 托管策略左侧的框并选择附加策略。
4. 选择信任关系，然后选择编辑信任关系。
5. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "codedeploy.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. 如果使用蓝/绿部署类型的 Amazon ECS 服务中的任务要求使用任务执行角色或任务角色覆盖，则必须将每个任务执行角色或任务角色覆盖的 `iam:PassRole` 权限作为内联策略添加到 CodeDeploy IAM 角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色 (p. 406) 和 Amazon ECS 任务角色 (p. 418)。

按照以下子步骤创建内联策略。

a. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
b. 在角色列表中搜索 `ecsCodeDeployRole`。如果该角色不存在，请使用上述过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
c. 在 Permissions policies (权限策略) 部分中，选择 Add inline policy (添加内联策略)。
d. 选择 JSON 选项卡，然后添加以下策略文本。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"
            ]
        }
    ]
}
```

Note

指定您的任务执行角色或任务角色覆盖的完整 ARN。

e. 选择查看策略
f. 对于名称，键入添加的策略的名称，然后选择创建策略。

Amazon ECS 服务 Auto Scaling IAM 角色

Application Auto Scaling 服务需要先获得描述 CloudWatch 警报和已注册服务的权限，以及代表您更新 Amazon ECS 服务的所需数量的权限，然后您才能将 服务 Auto Scaling 用于 Amazon ECS。这些权限由 服务 Auto Scaling IAM 角色 (`ecsAutoscaleRole`) 提供。

Note

IAM 用户还需要对 服务 Auto Scaling 的使用权限；服务 Auto Scaling 需要 IAM 权限 (p. 300)中描述了这些权限。如果一个 IAM 用户有权在 Amazon ECS 控制台中使用服务 Auto Scaling，创建 IAM 角色，并将 IAM 角色策略附加到这些角色，则该用户可自动创建此角色（包含在 Amazon ECS 控制台创建服务 (p.)或更新服务 (p. 326)工作流程中），然后将此角色用于任何其他服务（在控制台中或通过 AWS CLI 或开发工具包）。

您可使用以下过程检查并确定您的账户是否已有 服务 Auto Scaling IAM 角色。

`AmazonEC2ContainerServiceAutoscaleRole` 策略如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1456535218000",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeServices",
                "ecs:UpdateService"
            ],
            "Resource": [
```

```
                "*"
            ]
        },
        {
            "Sid": "Stmt1456535243000",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

在 IAM 控制台中检查 服务 Auto Scaling 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsAutoscaleRole`。如果该角色不存在，请使用以下过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions 选项卡。
5. 在托管策略部分中，确保将 AmazonEC2ContainerServiceAutoscaleRole 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS 服务角色。否则，请执行以下子步骤来附加策略。
   a. 选择 Attach policies (附加策略)。
   b. 要缩小要附加的可用策略的范围，请为筛选条件键入 `AmazonEC2ContainerServiceAutoscaleRole`。
   c. 选中 AmazonEC2ContainerAutoscaleRole 策略左侧的框并选择附加策略。
6. 选择信任关系，然后选择编辑信任关系。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "application-autoscaling.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建用于 服务 Auto Scaling 的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 在选择将使用此角色的服务部分中，选择 Elastic Container Service。
4. 在选择您的使用案例部分中，选择 Elastic Container Service Autoscale 和下一步：权限。
5. 对于添加标签（可选），输入要添加到 IAM 角色的任何键值标签。完成后，选择下一步：审核。
6. 在角色名称字段中，键入 `ecsAutoscaleRole` 以命名角色，然后选择创建角色以完成操作。

CloudWatch Events IAM 角色

CloudWatch Events 服务需要先获得代表您运行 Amazon ECS 任务的权限，然后您才能将 Amazon ECS 计划任务用于 CloudWatch Events 规则和目标。这些权限是由 CloudWatch Events IAM 角色 (`ecsEventsRole`) 提供的。

当您配置计划任务时，系统会在 AWS 管理控制台中自动为您创建 CloudWatch Events 角色。有关更多信息，请参阅排定的任务 (cron) (p. 263)。

`AmazonEC2ContainerServiceEventsRole` 策略如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

如果计划任务需要使用任务执行角色，或者需要使用任务角色覆盖，则必须为每个任务执行角色或对于 CloudWatch Events IAM 角色的任务角色覆盖添加 `iam:PassRole` 权限。有关任务执行角色的更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

> **Note**
>
> 指定您的任务执行角色或任务角色覆盖的完整 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"
            ]
        }
    ]
}
```

您可以使用以下过程检查您的账户是否已经拥有 CloudWatch Events IAM 角色，并根据需要手动创建。

在 IAM 控制台中检查 CloudWatch Events IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsEventsRole`。如果该角色不存在，请使用下一个过程创建该角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions。
5. 在权限策略部分中，确保将 AmazonEC2ContainerServiceEventsRole 托管策略附加到角色。如果附加该策略，则将正确配置 Amazon ECS 服务角色。否则，请执行以下子步骤来附加策略。

   a. 选择 Attach policies (附加策略)。
   b. 要缩小要附加的可用策略的范围，请为筛选条件键入 `AmazonEC2ContainerServiceEventsRole`。
   c. 选中 AmazonEC2ContainerServiceEventsRole 策略左侧的框并选择附加策略。
6. 选择信任关系，然后选择编辑信任关系。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建用于 CloudWatch Events 的 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 在选择受信任实体的类型部分，选择 Elastic Container Service。对于选择您的使用案例，选择 Elastic Container Service 任务。选择 Next: Permissions (下一步：权限)。
4. 在附加权限策略部分中，选择 AmazonEC2ContainerServiceEventsRole 策略并选择下一步：标签。
5. 在添加标签（可选）部分中，输入要与角色关联的任何标签，然后选择下一步：审核。
6. 对于角色名称，键入 `ecsEventsRole` 来命名角色，也可输入描述，然后选择创建角色。
7. 审核您的角色信息，然后选择创建角色。
8. 在角色列表中搜索 `ecsEventsRole` 并选择您刚刚创建的角色。
9. 选择信任关系，然后选择编辑信任关系。
10. 在策略文档窗口中，将现有信任关系替换为以下文本，然后选择更新信任策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

向 CloudWatch Events IAM 角色添加任务执行角色的权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在导航窗格中，选择 Policies、Create policy。
3. 选择 JSON，粘贴以下策略，然后选择 查看策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"
            ]
        }
    ]
}
```

4. 对于名称，键入 `AmazonECSEventsTaskExecutionRole`，也可输入描述，然后选择创建策略。
5. 在导航窗格中，选择 Roles。
6. 搜索角色列表，找到 `ecsEventsRole`，选择该角色来查看附加的策略。
7. 选择 Attach policy。
8. 在附加策略部分，选择 AmazonECSEventsTaskExecutionRole 策略，然后选择附加策略。

Amazon ECS 任务角色

Amazon ECS 需要先获得代表您调用 AWS API 的权限，然后您才能将 IAM 角色用于任务。这些权限由 Amazon ECS 任务角色提供。

您可以为每个任务定义创建一个需要获得调用 AWS API 的权限的任务 IAM 角色。只需创建一个定义任务应有权限的 IAM 策略，然后将该策略附加到使用 Amazon ECS 任务角色信任关系策略的角色。有关更多信息，请参阅为任务创建 IAM 角色和策略 (p. 420)。

Amazon ECS 任务角色信任关系如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

任务的 IAM 角色

借助 Amazon ECS 任务的 IAM 角色，您可以指定一个可由任务中的容器使用的 IAM 角色。应用程序必须使用 AWS 凭证签署其 AWS API 请求，并且此功能提供了一个管理凭证的策略以供应用程序使用，类似于 Amazon EC2 实例配置文件为 EC2 实例提供凭证的方式。您可以将 IAM 角色与 ECS 任务定义或 `RunTask` API 操作关联，而不是为容器创建和分配 AWS 凭证或使用 EC2 实例的角色。之后，任务的容器中的应用程序可以使用 AWS 开发工具包或 CLI 向授权的 AWS 服务发出 API 请求。

> **Important**
>
> 在您的容器实例上运行的容器可以访问通过 Amazon EC2 实例元数据服务器提供给容器实例配置文件的凭证。建议您将容器实例角色中的权限限制在 Amazon ECS 容器实例 IAM 角色 (p. 404)中显示的最低权限列表的范围内。
> 要防止使用 `awsvpc` 网络模式的任务中的容器访问提供给容器实例配置文件的凭证信息 (同时仍允许任务角色提供的权限)，请将代理配置文件中的 `ECS_AWSVPC_BLOCK_IMDS` 代理配置变量设置为 `true`，然后重新启动代理。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。
> 通过在容器实例上运行以下 iptables 命令，阻止使用 `bridge` 网络模式的任务中的容器访问提供给容器实例配置文件的凭证信息（同时仍允许任务角色提供的权限）。请注意，此命令不影响使用 `host` 或 `awsvpc` 网络模式的任务中的容器。有关更多信息，请参阅网络模式 (p. 52)。
>
> ```
> sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
> ```
>
> 您必须将此 iptables 规则保存到容器实例上，使其在重启后仍然可用。对于Amazon ECS-optimized AMI，请使用以下命令。对于其他操作系统，请参阅该操作系统的相关文档。
>
> - 对于Amazon ECS-optimized Amazon Linux 2 AMI：
>
> ```
> sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
> ```
>
> - 对于Amazon ECS-optimized Amazon Linux AMI：
>
> ```
> sudo service iptables save
> ```

您在任务定义中定义要使用的 IAM 角色，也可以在通过 `RunTask` API 操作手动运行任务时使用 `taskRoleArn` 覆盖。Amazon ECS 代理接收启动具有包含角色凭证的其他字段的任务的负载消息。Amazon ECS 代理将唯一的任务凭证 ID 设置为标识令牌并更新其内部凭证缓存，以便任务的标识令牌指向负载中接收的角色凭证。Amazon ECS 代理使用以下相关 URI 为属于此任务的所有容器填写 `Env` 对象（通过 docker inspect ***`container_id`*** 命令提供）中的 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 环境变量：/*`credential_provider_version`*/credentials?id=*`task_credential_id`*。

> **Note**
>
> 为任务指定 IAM 角色时，该任务的容器中的 AWS CLI 或其他开发工具包只使用该任务角色提供的 AWS 凭证，它们不再从容器实例继承任何 IAM 权限。

在容器中，可使用以下命令查询凭证：

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

输出:

```
{
    "AccessKeyId": "ACCESS_KEY_ID",
    "Expiration": "EXPIRATION_DATE",
    "RoleArn": "TASK_ROLE_ARN",
    "SecretAccessKey": "SECRET_ACCESS_KEY",
    "Token": "SECURITY_TOKEN_STRING"
}
```

如果容器实例使用的容器代理的版本至少为 1.11.0，并且使用的 AWS CLI 或开发工具包版本受支持，则开发工具包客户端将发现 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 变量可用，它将使用提供的凭证来调用 AWS API。有关更多信息，请参阅 在容器实例上启用任务 IAM 角色 (p. 420) 和 使用支持的 AWS 开发工具包 (p. 422)。

当使用凭证提供程序时，请求将本地记录在以下位置的主机容器实例中：`/var/log/ecs/audit.log.YYYY-MM-DD-HH`。有关更多信息，请参阅任务凭证审核日志的 IAM 角色 (p. 590)。

主题

- 使用任务的 IAM 角色的好处 (p. 420)
- 在容器实例上启用任务 IAM 角色 (p. 420)
- 为任务创建 IAM 角色和策略 (p. 420)
- 使用支持的 AWS 开发工具包 (p. 422)
- 为任务指定 IAM 角色 (p. 422)

使用任务的 IAM 角色的好处

- 凭证隔离：容器只能检索其所属的任务定义中定义的 IAM 角色的凭证；容器永远无法访问用于属于另一个任务的其他容器的凭证。
- 授权：未经授权的容器无法访问为其他任务定义的 IAM 角色凭证。
- 可审核性：可通过 CloudTrail 进行访问和事件日志记录以确保可追溯性审核。任务凭证具有连接到会话的 `taskArn` 的上下文，因此 CloudTrail 会显示哪个任务使用了哪个角色。

在容器实例上启用任务 IAM 角色

Amazon ECS 容器实例需要至少为 1.11.0 版的容器代理才能启用任务 IAM 角色；但建议使用最新的容器代理版本。有关检查您的代理版本并更新到最新版本的信息，请参阅更新 Amazon ECS 容器代理 (p. 211)。如果您使用Amazon ECS-optimized AMI，您的实例最低需要 1.11.0-1 版的 `ecs-init` 程序包。如果您的容器实例是从版本 2016.03.e 或更高版本启动的，这些实例将包含所需版本的容器代理和 `ecs-init`。有关更多信息，请参阅Amazon ECS-optimized AMI (p. 127)。

如果对容器实例使用的不是Amazon ECS-optimized AMI，请务必将 `--net=host` 选项添加到 docker run 命令，该命令会为所需配置启动代理和相应的代理配置变量（有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)）：

`ECS_ENABLE_TASK_IAM_ROLE=true`

为具有 `bridge` 和 `default` 网络模式的容器启用任务的 IAM 角色。

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

为具有 `host` 网络模式的容器启用任务的 IAM 角色。此变量仅在代理版本 1.12.0 和更高版本上受支持。

有关示例运行命令，请参阅手动更新 Amazon ECS 容器代理（适用于非Amazon ECS-Optimized AMI）(p. 215)。您还需要在容器实例上设置以下联网命令，以便任务中的容器可以检索其 AWS 凭证：

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --to-
ports 51679
```

您必须将这些 iptables 规则保存到容器实例，使其在重启后仍然可用。您可以使用 iptables-save 和 iptables-restore 命令保存 iptables 规则并在启动时还原它们。有关更多信息，请查阅您的特定操作系统文档。

为任务创建 IAM 角色和策略

您必须创建一个供任务使用的 IAM 策略，该策略指定您希望任务中的容器拥有的权限。您可通过多种方式创建新的 IAM 权限策略。您可以复制已具有您期望的一些功能的完整 AWS 托管策略，然后根据您的特定要求对其进行自定义。有关更多信息，请参阅 IAM 用户指南 中的创建新策略。

此外，您必须先创建一个供任务使用的角色，然后才能在任务定义中指定该角色。可以在 IAM 控制台中使用 Amazon Elastic Container Service Task Role (Amazon Elastic Container Service 任务角色) 服务角色创建该角色。然后，可以将您特定的 IAM 策略附加到该角色，该策略为任务中的容器提供所需的权限。以下过程说明如何执行此操作。

> **Note**
>
> 要查看此角色的信任关系，请参阅Amazon ECS 任务角色 (p. 418)。

如果您有多个需要 IAM 权限的任务定义或服务，则应考虑为每个特定的任务定义或服务创建一个具有所需最低权限的角色以便任务进行操作，以便您能够将为每个任务提供的访问权限降到最低。

为任务创建 IAM 策略

在此示例中，我们将创建一个策略以允许对 Amazon S3 存储桶进行只读访问。您可以将数据库凭证或其他机密信息存储在存储桶中，您的任务中的容器可以从存储桶中读取凭证并将它们加载到应用程序中。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中选择策略，然后选择创建策略。
3. 按照以下某个选项卡下的步骤操作，这些步骤介绍如何使用可视化编辑器或 JSON 编辑器。

Using the visual editor

1. 对于 Service，选择 S3。
2. 对于 Actions (操作)，展开 Read (读取) 选项，然后选择 GetObject。
3. 对于 Resources (资源)，选择 Add ARN (添加 ARN)，然后输入 Amazon S3 存储桶的完整 ARN。
4. 选择查看策略。
5. 在 Review policy (查看策略) 部分中，对于 Name (名称)，键入您自己的唯一名称，例如 `AmazonECSTaskS3BucketPolicy`。
6. 选择 Create policy (创建策略) 以完成。

Using the JSON editor

1. 在 Policy Document 字段中，粘贴要应用于任务的策略。以下示例授予对 *`my-task-secrets-bucket`* Amazon S3 存储桶的权限。您可以修改策略文档以满足您的特定需求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::my-task-secrets-bucket/*"
      ]
    }
  ]
}
```

2. 选择 Create policy (创建策略)。

为任务创建 IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在导航窗格中，选择 Roles 和 Create New Role。
3. 在选择角色类型部分中，为 Amazon Elastic Container Service Task Role (Amazon Elastic Container Service 任务角色) 服务角色选择选择。

   Note

   要查看此角色的信任关系，请参阅Amazon ECS 任务角色 (p. 418)。
4. 在附加策略部分中，选择要用于任务的策略（在此示例中为 `AmazonECSTaskS3BucketPolicy`），然后选择下一步。
5. 对于角色名称，为您的角色输入一个名称。在此示例中，键入 `AmazonECSTaskS3BucketRole` 来命名该角色，然后选择创建角色以完成。

使用支持的 AWS 开发工具包

2016 年 7 月 13 日将对任务的 IAM 角色的支持添加到 AWS 开发工具包中。您的任务中的容器必须使用在该日期或之后创建的 AWS 开发工具包版本。Linux 分发版本的程序包管理器中包含的 AWS 开发工具包可能不是最新的，无法支持此功能。

为了确保您使用的是受支持的开发工具包，请在构建容器时按照适用于 Amazon Web Services 的工具中针对您的首选开发工具包的安装说明操作以获取最新版本。

为任务指定 IAM 角色

创建一个角色并向该角色附加策略后，您可以运行带入该角色的任务。您可通过多种方式执行此操作：

- 在任务定义中为任务指定 IAM 角色。您可以创建新的任务定义或现有任务定义的新版本并指定您之前创建的角色。如果使用控制台创建任务定义，请在 Task Role (任务角色) 字段中选择 IAM 角色。如果使用 AWS CLI 或开发工具包，请使用 `taskRoleArn` 参数指定任务角色 ARN。有关更多信息，请参阅 创建任务定义 (p. 45)。

  Note

  如果要在 Amazon ECS 服务中使用 IAM 任务角色，则需要此选项。
- 在运行任务时指定 IAM 任务角色覆盖。您可以在运行任务时指定 IAM 任务角色覆盖。如果使用控制台运行任务，请选择高级选项，然后在 Task Role (任务角色) 字段中选择 IAM 角色。如果使用 AWS CLI 或开发工具包，请使用 `overrides` JSON 对象中的 `taskRoleArn` 参数指定任务角色 ARN。有关更多信息，请参阅 正在运行的任务 (p. 251)。

  Note

  除了运行任务和服务所需的标准 Amazon ECS 权限外，IAM 用户还需要 `iam:PassRole` 权限，以使用 IAM 角色执行任务。

故障排除 Amazon Elastic Container Service 身份和访问

使用以下信息可帮助您诊断和修复在使用 Amazon ECS 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 Amazon ECS 中执行操作 (p. 423)
- 我无权执行 iam:PassRole (p. 423)
- 我想要查看我的访问密钥 (p. 423)
- 我是管理员并希望允许其他人访问 Amazon ECS (p. 423)

- 我想要允许我的 AWS 账户之外的用户访问我的 Amazon ECS 资源 (p. 424)

我无权在 Amazon ECS 中执行操作

如果 AWS 管理控制台 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

当 `mateojackson` IAM 用户尝试使用控制台查看有关 *`widget`* 的详细信息，但不具有 `ecs:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecs:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `ecs:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

我无权执行 iam:PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。请求那个人更新您的策略，以便允许您将角色传递给 Amazon ECS。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon ECS 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

我想要查看我的访问密钥

创建 IAM 用户访问密钥之后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，甚至为了帮助找到您的规范用户 ID 也不能提供。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果您丢失了秘密访问密钥，则必须向您的 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南 中的管理访问密钥。

我是管理员并希望允许其他人访问 Amazon ECS

要允许其他人访问 Amazon ECS，您必须为需要访问权限的人员或应用程序创建 IAM 实体（用户或角色）。他们（它们）将使用该实体的凭证访问 AWS。然后，您必须将策略附加到实体，以便在 Amazon ECS 中为他们（它们）授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南 中的创建您的第一个 IAM 委托用户和组。

我想要允许我的 AWS 账户之外的用户访问我的 Amazon ECS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon ECS 是否支持这些功能，请参阅Amazon Elastic Container Service 如何与 IAM 协同工作 (p. 370)。
- 要了解如何向您拥有的 AWS 账户中的资源提供访问权限，请参阅 IAM 用户指南 中的对您拥有的 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何向第三方 AWS 账户提供对您的资源的访问权限，请参阅 IAM 用户指南 中的向第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南 中的向经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南 中的 IAM 角色和基于资源的策略有何不同。

使用 Amazon ECS 命令行界面

Amazon Elastic Container Service (Amazon ECS) 命令行界面 (CLI) 提供用于简化从本地开发环境创建、更新和监控集群和任务的高级命令。Amazon ECS CLI 支持 Docker Compose 文件，Docker Compose 是一个流行的开源规范，用于定义和运行多容器应用程序。在每日开发和测试过程中使用 ECS CLI，取代 AWS 管理控制台。

> **Important**
>
> 目前，最新版本的 Amazon ECS CLI 仅支持主要版本的 Docker Compose 文件语法（版本 1、2 和 3）。Compose 文件中指定的版本必须为字符串 `"1"`、`"1.0"`、`"2"`、`"2.0"`、`"3"` 或 `"3.0"`。Docker Compose 次要版本不受支持。

Amazon ECS CLI 的最新版本为 1.15.1。有关发行说明，请参阅更改日志。

> **Note**
>
> GitHub 上提供 Amazon ECS CLI 的源代码。我们鼓励您针对要包含的更改提交提取请求。但是，Amazon Web Services 当前不支持运行此软件的修改后副本。

主题

- 安装 Amazon ECS CLI (p. 425)
- 配置 Amazon ECS CLI (p. 430)
- 迁移配置文件 (p. 432)
- 教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群 (p. 433)
- 教程：使用 Amazon ECS CLI 创建包含 EC2 任务的集群 (p. 437)
- 教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 (p. 442)
- Amazon ECS 命令行参考 (p. 445)

安装 Amazon ECS CLI

按照以下说明操作以在 macOS、Linux 或 Windows 系统上安装 Amazon ECS CLI。

步骤 1：下载 Amazon ECS CLI

下载 Amazon ECS CLI 二进制文件。

- 对于 macOS：

```
sudo curl -o /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest
```

- 对于 Linux 系统：

```
sudo curl -o /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest
```

- 对于 Windows 系统：

  打开 Windows PowerShell 并运行以下命令：

```
PS C:\> New-Item 'C:\Program Files\Amazon\ECSCLI' -type
 directory
PS C:\> Invoke-WebRequest -OutFile 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe' https://
amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe
```

  Note

  如果您遇到权限问题，请确保您以管理员身份运行 PowerShell。

步骤 2：(可选) 验证 Amazon ECS CLI

要验证 Amazon ECS CLI 文件的有效性，您可以使用提供的 MD5 总和或 PGP 签名。以下部分中将介绍这两种方法。

使用 MD5 总和验证

使用提供的 MD5 总和验证已下载的二进制文件。

- 对于 macOS (比较两个输出字符串以验证它们是否匹配)：

```
curl -s https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest.md5 && md5 -
q /usr/local/bin/ecs-cli
```

- 对于 Linux 系统 (在输出字符串中查找 OK)：

```
echo "$(curl -s https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest.md5) /
usr/local/bin/ecs-cli" | md5sum -c -
```

- 对于 Windows 系统：

  打开 Windows PowerShell，并找到您下载的可执行文件的 md5 散列：

```
PS C:\> Get-FileHash ecs-cli.exe -Algorithm MD5
```

  将其与此 md5 散列进行比较：

```
PS C:\> Invoke-WebRequest -OutFile md5.txt https://
amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-
latest.md5
PS C:\> Get-Content md5.txt
```

使用 PGP 签名验证

Amazon ECS CLI 可执行文件是使用 PGP 签名进行加密签名的。您可以通过以下步骤来使用 GnuPG 工具验证签名。

1. 下载并安装 GnuPG。有关更多信息，请参阅 GnuPG 网站。

   - 对于 macOS，我们建议使用 Homebrew。按照其网站中的说明安装 Homebrew。有关更多信息，请参阅 Homebrew。安装 Homebrew 后，从您的 macOS 终端使用以下命令：

```
brew install gnupg
```

- 对于 Linux 系统，使用您的 Linux 风格的程序包管理器安装 gpg。
- 对于 Windows 系统，下载并使用 GnuPG 网站上的 Windows 简单安装程序。有关更多信息，请参阅 GnuPG 下载。

2. 检索 Amazon ECS PGP 公钥。您可以使用命令检索，也可以手动创建此密钥然后导入。

   a. 选项 1：使用以下命令检索此密钥。

```
gpg --keyserver hkp://keys.gnupg.net --recv BCE9D9A42D51784F
```

   b. 选项 2：创建包含 Amazon ECS PGP 公钥的以下内容的文件，然后导入此文件：


```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
```

AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt

```
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

作为参考的 Amazon ECS PGP 公钥的详细信息：

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

使用以下命令导入 Amazon ECS PGP 公钥。

```
gpg --import <public_key_filename>
```

3. 下载 Amazon ECS CLI 签名。签名是存储在扩展名为 `.asc` 的文件中的 ASCII 分离 PGP 签名。此签名文件的名称与其对应可执行文件的名称相同，追加了 `.asc`。

   - 对于 macOS 系统：

     ```
     curl -o ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-
     latest.asc
     ```

   - 对于 Linux 系统：

     ```
     curl -o ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-
     latest.asc
     ```

   - 对于 Windows 系统：

     ```
     PS C:\> Invoke-WebRequest -OutFile ecs-cli.asc https://amazon-ecs-
     cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe.asc
     ```

4. 验证签名。

   - 对于 macOS 和 Linux 系统：

     ```
     gpg --verify ecs-cli.asc /usr/local/bin/ecs-cli
     ```

   - 对于 Windows 系统：

     ```
     PS C:\> gpg --verify ecs-cli.asc 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe'
     ```

   预期输出：

```
gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
gpg:                using RSA key DE3CBD61ADAF8B8E
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
```

> **Important**
>
> 输出中的警告是预料中的，没有问题。它出现是因为您的个人 PGP 密钥（如果您有）和 Amazon ECS PGP 密钥之间没有信任链。有关更多信息，请参阅信任 Web。

步骤 3：将执行权限应用于二进制文件

将执行权限应用于二进制文件。

- 对于 macOS 和 Linux 系统：

```
sudo chmod +x /usr/local/bin/ecs-cli
```

- 对于 Windows 系统：

  编辑环境变量并将 `C:\Program Files\Amazon\ECSCLI` 添加到 `PATH` 变量字段中，使用分号与现有条目分隔开来。例如：

```
PS C:\> C:\existing\path;C:\Program Files\Amazon\ECSCLI
```

  重新启动 PowerShell (或命令提示符)，以便更改生效。

  > **Note**
  >
  > 在设置 `PATH` 变量后，可以从 Windows PowerShell 或命令提示符使用 Amazon ECS CLI。

步骤 4：完成安装

验证 CLI 是否正常运行。

```
ecs-cli --version
```

继续执行配置 Amazon ECS CLI (p. 430)。

> **Important**
>
> 您必须使用 AWS 凭证、AWS 区域和 Amazon ECS 集群名称配置 Amazon ECS CLI，然后才能使用它。

配置 Amazon ECS CLI

Amazon ECS CLI 需要一些基本配置信息（例如，您的 AWS 凭证、要在其中创建集群的 AWS 区域以及要使用的 Amazon ECS 集群的名称），然后才能使用它。在 macOS 和 Linux 系统中，配置信息存储在 `~/.ecs` 目录中；在 Windows 系统中，则存储在 `C:\Users\<username>\AppData\local\ecs` 中。

配置 Amazon ECS CLI

1. 使用以下命令设置 CLI 配置文件，将 *profile_name* 替换为所需要的配置文件名称，将 *$AWS_ACCESS_KEY_ID* 和 *$AWS_SECRET_ACCESS_KEY* 环境变量替换为您的 AWS 凭证。

```
ecs-cli configure profile --profile-name profile_name --access-key $AWS_ACCESS_KEY_ID
 --secret-key $AWS_SECRET_ACCESS_KEY
```

2. 使用以下命令完成配置，将 *launch_type* 替换为您想使用的默认任务启动类型，将 *region_name* 替换为所需要的 AWS 区域，将 *cluster_name* 替换为要使用的现有 Amazon ECS 集群或新集群的名称，将 *configuration_name* 替换为要为此配置提供的名称。

```
ecs-cli configure --cluster cluster_name --default-launch-type launch_type --
region region_name --config-name configuration_name
```

安装并配置 CLI 后，您可以尝试 教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群 (p. 433)。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考 。

配置文件

Amazon ECS CLI 支持使用 ecs-cli configure profile 命令将多组 AWS 凭证配置为命名配置文件。使用 ecs-cli configure profile default 命令可以设置默认配置文件。使用 `--ecs-profile` 标志运行需要凭证的 Amazon ECS CLI 命令时，可以引用这些配置文件，否则将使用默认配置文件。

有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考 。

集群配置

集群配置是用于描述 Amazon ECS 集群的一组字段，包括集群名称和区域。使用 ecs-cli configure default 命令可以设置默认集群配置。Amazon ECS CLI 支持使用 `--config-name` 选项配置多个命名集群配置。

有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考 。

优先顺序

在 Amazon ECS CLI 命令中传递凭证和区域有多种方法。以下是每项的优先顺序。

凭证的优先顺序是：

1. Amazon ECS CLI 配置文件标志：
   a. ECS 配置文件 (`--ecs-profile`)
   b. AWS 配置文件 (`--aws-profile`)
2. 环境变量:
   a. `ECS_PROFILE`
   b. `AWS_PROFILE`
   c. `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 `AWS_SESSION_TOKEN`
3. ECS config -- 尝试从默认 ECS 配置文件中获取凭证。
4. 默认 AWS 配置文件 - 尝试从 —配置文件名称中使用凭证 （ `aws_access_key_id`、`aws_secret_access_key` ）或 `assume_role` （ `role_arn`、`source_profile` ）AWS 。
   a. `AWS_DEFAULT_PROFILE` 环境变量 (默认为 `default`).
5. EC2 实例角色

区域的优先顺序是：

1. Amazon ECS CLI 标志：
   a. 区域标志 (`--region`)
   b. 集群配置标志 (`--cluster-config`)
2. ECS config - 尝试从默认 ECS 配置文件中获取区域。
3. 环境变量 — 尝试从以下环境变量中获取区域：
   a. `AWS_REGION`
   b. `AWS_DEFAULT_REGION`
4. AWS 配置文件 - 尝试从 AWS 配置文件名称中使用区域：
   a. `AWS_PROFILE` 环境变量
   b. `AWS_DEFAULT_PROFILE` 环境变量 (默认为 `default`)

迁移配置文件

在最新版本 (v1.0.0) 中，配置 Amazon ECS CLI 的过程发生了显著变化，以便添加新的功能。推出了迁移命令，可将较旧的 (v0.6.6 及更低版本) 配置文件转换为当前格式。旧配置文件已弃用，因此我们建议将您的配置转换为最新格式，以便利用新的功能。在 v1.0.0 中，新的 YAML 格式配置文件中推出的配置相关更改和新功能包括：

- 将凭证和集群相关的配置信息拆分为两个单独的文件。凭证信息存储在 `~/.ecs/credentials` 中，集群配置信息存储在 `~/.ecs/config` 中。
- 配置文件为 YAML 格式。
- 支持存储多个命名配置。
- 弃用字段 `compose-service-name-prefix`（此名称用于创建服务 `<compose_service_name_prefix> + <project_name>`）。此字段仍然可以配置。但是，如果未配置，则不再分配默认值。对于 Amazon ECS CLI v0.6.6 和之前的版本，默认值为 `ecscompose-service-`。
- 删除字段 `compose-project-name-prefix`（此名称用于创建任务定义 `<compose_project_name_prefix> + <project_name>`）。Amazon ECS CLI v1.0.0 及更高版本仍然可以读取旧配置文件；因此，如果显示此字段，则仍然会读取和使用此字段。但是，v1.0.0 以上的版本不支持使用 `ecs-cli configure` 命令配置此字段，如果将此字段手动添加到 v1.0.0 以上版本的配置文件，将导致 Amazon ECS CLI 引发错误。
- 字段 `cfn-stack-name-prefix` (此名称用于创建 CFN 堆栈 `<cfn_stack_name_prefix> + <cluster_name>`) 已更改为 `cfn-stack-name`。除了指定前缀，可以配置 CloudFormation 模板的确切名称。
- Amazon ECS CLI v0.6.6 及之前的版本允许使用系统上的 `~/.aws/credentials` 文件中的命名 AWS 配置文件配置凭证。此功能已删除。但增加了一个新的标志 `--aws-profile`，它允许在需要凭证的所有命令中引用 AWS 配置文件内联。

> **Note**
>
> `--project-name` 标志可用于设置项目名称。

将较旧的配置文件迁移到 v1.0.0+ 格式

虽然 Amazon ECS CLI 的所有版本都支持读取较旧的配置文件格式，但要利用某些新功能，例如使用多个命名集群配置文件，则需要升级到新格式。使用 `ecs-cli configure migrate` 命令可以轻松地将旧配置文件迁移到新格式。该命令可以将 `~/.ecs/config` 中存储的旧格式配置信息转换为一对新格式文件，并在此过程中覆盖您的旧配置文件。

运行 `ecs-cli configure migrate` 命令时，将会出现旧配置文件的警告消息，以及新配置文件的预览。用户在迁移前需要进行确认。如果使用 `--force` 标志，则不会显示警告消息，且迁移前无需进行确认。如果旧文件中使用 `cfn-stack-name-prefix`，则 `cfn-stack-name` 将在新文件中存储为 `<cfn_stack_name_prefix> + <cluster_name>`。

有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考。

教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群

本教程介绍如何通过使用 Fargate 启动类型的任务来设置集群和部署服务。

先决条件

完成以下先决任务：

- 设置 AWS 账户。
- 安装 Amazon ECS CLI。有关更多信息，请参阅安装 Amazon ECS CLI (p. 425)。
- 安装和配置 AWS CLI。有关更多信息，请参阅 AWS 命令行界面。

步骤 1：创建任务执行 IAM 角色

Amazon ECS 需要获得权限，Fargate 任务才能在 CloudWatch 中存储日志。这些权限包含在任务执行 IAM 角色中。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

使用 AWS CLI 创建任务执行 IAM 角色

1. 使用以下内容创建名为 `task-execution-assume-role.json` 的文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 创建任务执行角色：

```
aws iam --region us-west-2 create-role --role-name ecsTaskExecutionRole --assume-role-
policy-document file://task-execution-assume-role.json
```

3. 附加任务执行角色策略：

```
aws iam --region us-west-2 attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

步骤 2：配置 Amazon ECS CLI

Amazon ECS CLI 需要获得凭证才能代表您发出 API 请求。它可以从环境变量、AWS 配置文件或 Amazon ECS 配置文件中提取凭证。有关更多信息，请参阅 配置 Amazon ECS CLI (p. 430)。

创建 Amazon ECS CLI 配置

1. 创建一个集群配置，该配置定义要使用的 AWS 区域、资源创建前缀以及要在 Amazon ECS CLI 中使用的集群名称：

```
ecs-cli configure --cluster tutorial --region us-west-2 --default-launch-type FARGATE
 --config-name tutorial
```

2. 使用访问密钥和私有密钥创建 CLI 配置文件：

```
ecs-cli configure profile --access-key AWS_ACCESS_KEY_ID --secret-
key AWS_SECRET_ACCESS_KEY --profile-name tutorial
```

> **Note**
>
> 如果这是您首次配置 Amazon ECS CLI，则这些配置将被标记为默认值。如果这不是您首次配置 Amazon ECS CLI，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考，以将其设置为默认配置和配置文件。

步骤 3：创建集群和安全组

创建 ECS 集群和安全组

1. 使用 ecs-cli up 命令创建 Amazon ECS 集群。由于您在集群配置中指定 Fargate 作为您的默认启动类型，因此该命令将创建一个空集群和配置有两个公有子网的一个 VPC。

```
ecs-cli up
```

> **Note**
>
> 创建您的资源时，完成此命令可能需要几分钟的时间。记下创建的 VPC 和子网 ID，以后将会用到它们。

2. 利用之前的输出中的 VPC ID，使用 AWS CLI 创建一个安全组：

```
aws ec2 create-security-group --group-name "my-sg" --description "My security group" --
vpc-id "VPC_ID"
```

3. 使用 AWS CLI 添加一个安全组规则，以允许端口 80 的入站访问：

```
aws ec2 authorize-security-group-ingress --group-id "security_group_id" --protocol tcp
 --port 80 --cidr 0.0.0.0/0
```

步骤 4：创建复合文件

在此步骤中，创建一个简单的 Docker 复合文件，此文件将创建一个简单的 PHP Web 应用程序。此时，Amazon ECS CLI 支持 Docker 复合文件语法版本 1、2 和 3。本教程使用 Docker compose v3。

以下是复合文件，您可以将它称作 `docker-compose.yml`。web 容器将开放端口 80 以便将流量传入 Web 服务器。它还会配置容器日志，以转到之前创建的 CloudWatch 日志组。这是针对 Fargate 任务建议的最佳实践。

```
version: '3'
services:
  web:
    image: amazon/amazon-ecs-sample
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial
        awslogs-region: us-west-2
        awslogs-stream-prefix: web
```

除了 Docker 复合信息，您还必须为服务指定某些特定于 Amazon ECS 的参数。使用上一步中的 VPC、子网和安全组 ID，创建一个名为 `ecs-params.yml` 且包含以下内容的文件：

```
version: 1
task_definition:
  task_execution_role: ecsTaskExecutionRole
  ecs_network_mode: awsvpc
  task_size:
    mem_limit: 0.5GB
    cpu_limit: 256
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - "subnet ID 1"
        - "subnet ID 2"
      security_groups:
        - "security group ID"
      assign_public_ip: ENABLED
```

步骤 5：将复合文件部署到集群

在创建复合文件后，您可以使用 ecs-cli compose service up 将其部署到集群。默认情况下，此命令将查找当前目录中名为 `docker-compose.yml` 和 `ecs-params.yml` 的文件；您可以使用 `--file` 选项指定不同的 Docker 复合文件，并使用 `--ecs-params` 选项指定不同的 ECS Params 文件。默认情况下，通过此命令创建的资源具有标题中的当前目录，但您可以使用 `--project-name` 选项覆盖该目录。`--create-log-groups` 选项将为容器日志创建 CloudWatch 日志组。

```
ecs-cli compose --project-name tutorial service up --create-log-groups --cluster-
config tutorial
```

步骤 6：查看集群上正在运行的容器

在部署复合文件后，您可以使用 ecs-cli compose service ps 查看正在服务中运行的容器。

```
ecs-cli compose --project-name tutorial service ps --cluster-config tutorial
```

输出:

```
Name                                          State    Ports
 TaskDefinition  Health
```

```
tutorial/0c2862e6e39e4eff92ca3e4f843c5b9a/web  RUNNING  34.222.202.55:80->80/tcp
 tutorial:1      UNKNOWN
```

在上述示例中，您可以查看复合文件中的 web 容器以及 Web 服务器的 IP 地址和端口。如果将 Web 浏览器指向此地址，则应看到该 PHP Web 应用程序。此外，输出中还显示容器的 task-id 值。复制任务 ID；您将在下一步中使用它。

步骤 7：查看容器日志

查看任务日志：

```
ecs-cli logs --task-id 0c2862e6e39e4eff92ca3e4f843c5b9a --follow --cluster-config tutorial
```

> Note
>
> --follow 选项告知 Amazon ECS CLI 持续轮询日志。

步骤 8：在集群上扩展任务

您可以使用 ecs-cli compose service scale 向上扩展任务计数，以增加应用程序的实例数量。在此示例中，正在运行的应用程序计数增至两个。

```
ecs-cli compose --project-name tutorial service scale 2 --cluster-config tutorial
```

现在，您的集群中应拥有两个以上的容器：

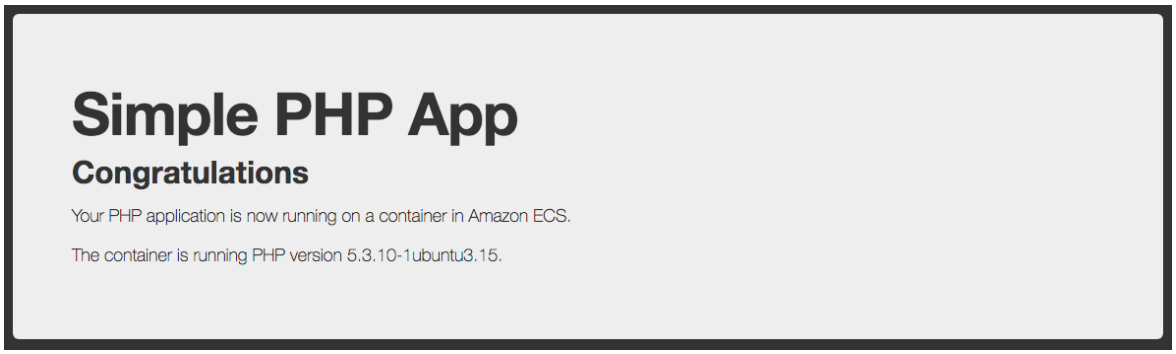
```
ecs-cli compose --project-name tutorial service ps --cluster-config tutorial
```

输出:

```
Name                                          State    Ports
 TaskDefinition  Health
tutorial/0c2862e6e39e4eff92ca3e4f843c5b9a/web  RUNNING  34.222.202.55:80->80/tcp
 tutorial:1      UNKNOWN
tutorial/d9fbbc931d2e47ae928fcf433041648f/web  RUNNING  34.220.230.191:80->80/tcp
 tutorial:1      UNKNOWN
```

步骤 9：（可选）查看您的 Web 应用程序

在 Web 浏览器中输入任务的 IP 地址，您应看到一个显示 Simple PHP App Web 应用程序的网页。



步骤 10：清除

在结束此教程后，应清除您的资源以便这些资源不再产生任何费用。首先，删除服务，以便它停止现有容器且不尝试运行任何其他任务。

```
ecs-cli compose --project-name tutorial service down --cluster-config tutorial
```

现在，关闭您的集群，这将清除先前使用 ecs-cli up 创建的资源。

```
ecs-cli down --force --cluster-config tutorial
```

教程：使用 Amazon ECS CLI 创建包含 EC2 任务的集群

本教程说明了如何使用 EC2 启动类型设置集群和部署任务。

先决条件

完成以下先决任务：

- 设置 AWS 账户。
- 安装 Amazon ECS CLI。有关更多信息，请参阅安装 Amazon ECS CLI (p. 425)。
- 安装和配置 AWS CLI。有关更多信息，请参阅 AWS 命令行界面。

步骤 1：配置 Amazon ECS CLI

您必须先安装并配置 Amazon ECS CLI，然后才能启动此教程。有关更多信息，请参阅安装 Amazon ECS CLI (p. 425)。

Amazon ECS CLI 需要获得凭证才能代表您发出 API 请求。它可以从环境变量、AWS 配置文件或 Amazon ECS 配置文件中提取凭证。有关更多信息，请参阅 配置 Amazon ECS CLI (p. 430)。

创建 Amazon ECS CLI 配置

1. 创建集群配置：

```
ecs-cli configure --cluster ec2-tutorial --region us-east-1 --default-launch-type EC2
 --config-name ec2-tutorial
```

2. 使用您的访问密钥和私有密钥创建配置文件：

```
ecs-cli configure profile --access-key AWS_ACCESS_KEY_ID --secret-
key AWS_SECRET_ACCESS_KEY --profile-name tutorial-profile
```

> **Note**
>
> 如果这是您首次配置 Amazon ECS CLI，则这些配置将被标记为默认值。如果这不是您首次配置 Amazon ECS CLI，请参阅 ecs-cli configure default (p. 449) 和 ecs-cli configure profile default (p. 451)，以将其设置为默认配置和配置文件。

步骤 2：创建您的集群

您应执行的首个操作是创建 Amazon ECS 容器实例的集群，您可使用 ecs-cli up 命令在该集群上启动容器。虽然您在使用此命令配置集群时有许多选项可供选择，但大多数选项是可选的。在此示例中，您创建一个包含两个 `t2.medium` 容器实例的简单集群，这两个实例使用 *`id_rsa`* 密钥对进行 SSH 访问（此处替换您自己的密钥对）。

默认情况下，为容器实例创建的安全组将为入站流量打开端口 80。您可以使用 `--port` 选项指定要打开的其他端口，或者，如果有更复杂的安全组要求，则可指定现有安全组使用 `--security-group` 选项。

```
ecs-cli up --keypair id_rsa --capability-iam --size 2 --instance-type t2.medium --cluster-
config ec2-tutorial --ecs-profile tutorial-profile
```

创建您的资源时，完成此命令可能需要几分钟的时间。现在您已拥有集群，可以创建 Docker 复合文件并部署该文件。

步骤 3：创建复合文件

在此步骤中，创建一个简单的 Docker 复合文件，此文件将创建包含 Web 服务器和 MySQL 数据库的 WordPress 应用程序。此时，Amazon ECS CLI 支持 Docker 复合文件语法版本 1、2 和 3。提供了 Docker Compose 版本 2 和 3 的示例。

Amazon ECS CLI 的复合文件中支持以下参数：

- `cap_add` (对使用 Fargate 启动类型的任务无效)
- `cap_drop` (对使用 Fargate 启动类型的任务无效)
- `command`
- `cpu_shares`

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `cpu_shares`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。
- `devices` (对使用 Fargate 启动类型的任务无效)
- `dns`
- `dns_search`
- `entrypoint`
- `environment`：如果复合文件中未指定环境变量值，但该值存在于 shell 环境中，则 shell 环境变量值将传递到为任何关联的任务或服务创建的任务定义。

  > **Important**
  >
  > 建议不要对敏感信息 (如凭证数据) 使用纯文本环境变量。
- `env_file`

  > **Important**
  >
  > 建议不要对敏感信息 (如凭证数据) 使用纯文本环境变量。
- `extra_hosts`
- `healthcheck`（仅限 Compose 文件版本 3）

  > **Note**
  >
  > 使用复合文件时不支持 `start_period` 字段。要指定 `start_period`，请使用 `ecs-params.yml`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。

- `hostname`
- `image`
- `labels`
- `links` (对使用 Fargate 启动类型的任务无效)
- `log_driver`（仅限 Compose 文件版本 1）
- `log_opt`（仅限 Compose 文件版本 1）
- `logging`（Compose 文件版本 2 和 3）
  - `driver`
  - `options`
- `mem_limit` (以字节为单位)

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `mem_limit`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。
- `mem_reservation` (以字节为单位)

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `mem_reservation`。有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。
- `ports`
- `privileged` (对使用 Fargate 启动类型的任务无效)
- `read_only`
- `security_opt`
- `shm_size`（仅限 Compose 文件版本 1 和 2，并且对使用 Fargate 启动类型的任务无效）
- `tmpfs` (对使用 Fargate 启动类型的任务无效)
- `ulimits`
- `user`
- `volumes`
- `volumes_from`（仅限 Compose 文件版本 1 和 2）
- `working_dir`

> **Important**
>
> 此时不支持 `build` 指令。

有关 Docker 复合文件语法的更多信息，请参阅 Docker 文档中的复合文件参考。

以下是复合文件，您可以对它称作 `docker-compose.yml`。每个容器均有 100 个 CPU 单位和 500 MiB 内存。`wordpress` 容器将为容器实例开放端口 80 以便将流量传入 Web 服务器。还将定义容器的日志记录配置。

示例 1：Docker Compose 版本 2

```
version: '2'
services:
  wordpress:
    image: wordpress
    cpu_shares: 100
    mem_limit: 524288000
```

```
    ports:
      - "80:80"
    links:
      - mysql
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial-wordpress
        awslogs-region: us-east-1
        awslogs-stream-prefix: wordpress
  mysql:
    image: mysql:5.7
    cpu_shares: 100
    mem_limit: 524288000
    environment:
      MYSQL_ROOT_PASSWORD: password
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial-mysql
        awslogs-region: us-east-1
        awslogs-stream-prefix: mysql
```

示例 2：Docker Compose 版本 3

```
version: '3'
services:
  wordpress:
    image: wordpress
    ports:
      - "80:80"
    links:
      - mysql
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial-wordpress
        awslogs-region: us-east-1
        awslogs-stream-prefix: wordpress
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: password
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial-mysql
        awslogs-region: us-east-1
        awslogs-stream-prefix: mysql
```

当使用 Docker Compose 版本 3 格式时，必须分别指定 CPU 和内存规格。使用以下内容创建名为 `ecs-params.yml` 的文件：

```
version: 1
task_definition:
  services:
    wordpress:
      cpu_shares: 100
      mem_limit: 524288000
    mysql:
      cpu_shares: 100
      mem_limit: 524288000
```

步骤 4：将复合文件部署到集群

在创建复合文件后，您可以使用 ecs-cli compose up 命令将其部署到集群。默认情况下，此命令将在当前目录中查找一个名为 `docker-compose.yml` 的复合文件以及一个名为 `ecs-params.yml` 的可选 ECS 参数文件，但您可以使用 `--file` 选项指定另一个文件。默认情况下，通过此命令创建的资源具有标题中的当前目录，但您可以使用 `--project-name` *`project_name`* 选项覆盖该目录。`--create-log-groups` 选项将为容器日志创建 CloudWatch 日志组。

```
ecs-cli compose up --create-log-groups --cluster-config ec2-tutorial --ecs-
profile tutorial-profile
```

步骤 5：查看集群上正在运行的容器

在部署复合文件后，您可以使用 ecs-cli ps 命令查看正在集群上运行的容器。

```
ecs-cli ps --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

输出:

```
Name                                                        State    Ports
  TaskDefinition  Health
ec2-tutorial/4a5c3162979b4df6b44d2f4e6b1dd3e5/wordpress  RUNNING  18.205.157.201:80->80/tcp
  ecscompose:3    UNKNOWN
ec2-tutorial/4a5c3162979b4df6b44d2f4e6b1dd3e5/mysql      RUNNING
  ecscompose:3    UNKNOWN
```

在上述示例中，您可以查看复合文件中的 `wordpress` 和 `mysql` 容器以及 Web 服务器的 IP 地址和端口。如果将 Web 浏览器指向该地址，应会显示 WordPress 安装向导。

步骤 6：在集群上扩展任务

您可以使用 ecs-cli compose scale 命令向上扩展任务计数以便应用程序有更多实例。在此示例中，您可以将应用程序的计数增至两个。

```
ecs-cli compose scale 2 --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

现在，您的集群中应拥有两个以上的容器：

```
ecs-cli ps --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

输出:

```
Name                                                 State    Ports
  TaskDefinition  Health
ec2-tutorial/4a5c3162979b4df6b44d2f4e6b1dd3e5/wordpress  RUNNING  18.205.157.201:80->80/tcp
  ecscompose:3    UNKNOWN
ec2-tutorial/4a5c3162979b4df6b44d2f4e6b1dd3e5/mysql      RUNNING
  ecscompose:3    UNKNOWN
ec2-tutorial/da60d89e74474a2691d26ef5f9ba5a16/mysql      RUNNING
  ecscompose:3    UNKNOWN
ec2-tutorial/da60d89e74474a2691d26ef5f9ba5a16/wordpress  RUNNING  34.232.105.149:80->80/tcp
  ecscompose:3    UNKNOWN
```

步骤 7：从复合文件创建 ECS 服务

现在，您已了解您的容器运行正常，可以确保容器在发生故障或停止时被替换。可以使用 ecs-cli compose service up 命令从复合文件创建服务来执行此操作。此命令从最新复合文件创建任务定义（如果任务定义不存在），并使用此任务定义创建预期计数为 1 的 ECS 服务。

在启动您的服务之前，请使用 ecs-cli compose down 命令从复合文件停止容器，以便有可使用的空集群。

```
ecs-cli compose down --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

现在，您可以创建服务。

```
ecs-cli compose service up --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

步骤 8：清除

在结束此教程后，应清除您的资源以便这些资源不再产生任何费用。首先，删除服务，以便它停止现有容器且不尝试运行任何其他任务。

```
ecs-cli compose service rm --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

现在，关闭您的集群，这将清除先前使用 ecs-cli up 创建的资源。

```
ecs-cli down --force --cluster-config ec2-tutorial --ecs-profile tutorial-profile
```

教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务

本教程介绍创建配置为使用服务发现的 Amazon ECS 服务的简单演练。许多服务发现配置值可使用 ECS 参数文件或标志进行指定。在使用标志时，它们优先于 ECS 参数文件（如果同时存在）。在使用 Amazon ECS CLI 时，复合项目名称用作 ECS 服务的名称。

先决条件

在继续本教程前，您应当已经满足下列先决条件：

- 设置 AWS 账户。
- 安装 Amazon ECS CLI。有关更多信息，请参阅安装 Amazon ECS CLI (p. 425)。

配置 Amazon ECS CLI

您必须先安装并配置 Amazon ECS CLI，然后才能启动此教程。有关更多信息，请参阅安装 Amazon ECS CLI (p. 425)。

Amazon ECS CLI 需要获得凭证才能代表您发出 API 请求。它可以从环境变量、AWS 配置文件或 Amazon ECS 配置文件中提取凭证。有关更多信息，请参阅 配置 Amazon ECS CLI (p. 430)。

创建 Amazon ECS CLI 配置

1. 创建集群配置：

```
ecs-cli configure --cluster ec2-tutorial --region us-east-1 --default-launch-type EC2
 --config-name ec2-tutorial
```

2. 使用您的访问密钥和私有密钥创建配置文件：

```
ecs-cli configure profile --access-key AWS_ACCESS_KEY_ID --secret-
key AWS_SECRET_ACCESS_KEY --profile-name ec2-tutorial
```

> **Note**
>
> 如果这是您首次配置 Amazon ECS CLI，则这些配置将被标记为默认值。如果这不是您首次配置 Amazon ECS CLI，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 命令行参考，以将其设置为默认配置和配置文件。

创建一个配置为使用服务发现的 Amazon ECS 服务

按照以下步骤，使用 Amazon ECS CLI 创建配置为使用服务发现的 Amazon ECS 服务。

创建一个配置为使用服务发现的 Amazon ECS 服务

1. 创建一个名为 `backend` 的 Amazon ECS 服务并在 VPC 中创建一个名为 `tutorial` 的私有 DNS 命名空间。在此示例中，任务使用 `awsvpc` 网络模式，因此无需 `container_name` 和 `container_port` 值。

```
ecs-cli compose --project-name backend service up --private-dns-namespace tutorial --
vpc vpc-04deee8176dce7d7d --enable-service-discovery
```

输出:

```
INFO[0001] Using ECS task definition                     TaskDefinition="backend:1"
INFO[0002] Waiting for the private DNS namespace to be created...
INFO[0002] Cloudformation stack status                   stackStatus=CREATE_IN_PROGRESS
WARN[0033] Defaulting DNS Type to A because network mode was awsvpc
INFO[0033] Waiting for the Service Discovery Service to be created...
INFO[0034] Cloudformation stack status                   stackStatus=CREATE_IN_PROGRESS
INFO[0065] Created an ECS service                        service=backend
 taskDefinition="backend:1"
INFO[0066] Updated ECS service successfully              desiredCount=1
 serviceName=backend
INFO[0081] (service backend) has started 1 tasks: (task 824b5a76-8f9c-4beb-
a64b-6904e320630e).  timestamp="2018-09-12 00:00:26 +0000 UTC"
INFO[0157] Service status                                desiredCount=1 runningCount=1
 serviceName=backend
INFO[0157] ECS Service has reached a stable state        desiredCount=1 runningCount=1
 serviceName=backend
```

2. 在同一私有 DNS 命名空间中创建另一个名为 `frontend` 的服务。由于此命名空间已存在，Amazon ECS CLI 会使用它而不是创建一个新的。

```
ecs-cli compose --project-name frontend service up --private-dns-namespace tutorial --
vpc vpc-04deee8176dce7d7d --enable-service-discovery
```

输出:

```
INFO[0001] Using ECS task definition                     TaskDefinition="frontend:1"
INFO[0002] Using existing namespace ns-kvhnzhb5vxplfmls
```

```
WARN[0033] Defaulting DNS Type to A because network mode was awsvpc
INFO[0033] Waiting for the Service Discovery Service to be created...
INFO[0034] Cloudformation stack status                    stackStatus=CREATE_IN_PROGRESS
INFO[0065] Created an ECS service                         service=frontend
 taskDefinition="frontend:1"
INFO[0066] Updated ECS service successfully               desiredCount=1
 serviceName=frontend
INFO[0081] (service frontend) has started 1 tasks: (task 824b5a76-8f9c-4beb-
a64b-6904e320630e).  timestamp="2018-09-12 00:00:26 +0000 UTC"
INFO[0157] Service status                                 desiredCount=1 runningCount=1
 serviceName=frontend
INFO[0157] ECS Service has reached a stable state         desiredCount=1 runningCount=1
 serviceName=frontend
```

3. 验证这两个服务是否能够在 VPC 中使用 DNS 发现对方。DNS 主机名使用以下格式：`<service_discovery_service_name>.<service_discovery_namespace>`。在此示例中，`frontend` 服务可在 `frontend.tutorial` 处发现，`backend` 服务可在 `backend.tutorial` 处发现。由于这些是私有 DNS 命名空间，这些 DNS 名称仅在指定的 VPC 中时可解析。
4. 要更新服务发现设置，请更新 `frontend` 服务的设置。可更新的值是 DNS TTL 和运行状况检查自定义配置失败阈值的值。

```
ecs-cli compose --project-name frontend service up --update-service-discovery --dns-
type SRV --dns-ttl 120 --healthcheck-custom-config-failure-threshold 2
```

输出:

```
INFO[0001] Using ECS task definition                      TaskDefinition="frontend:1"
INFO[0001] Updated ECS service successfully               desiredCount=1
 serviceName=frontend
INFO[0001] Service status                                 desiredCount=1 runningCount=1
 serviceName=frontend
INFO[0001] ECS Service has reached a stable state         desiredCount=1 runningCount=1
 serviceName=frontend
INFO[0002] Waiting for your Service Discovery resources to be updated...
INFO[0002] Cloudformation stack status                    stackStatus=UPDATE_IN_PROGRESS
```

5. 要清除，请删除 Amazon ECS 服务和服务发现资源。在删除 `frontend` 服务后，Amazon ECS CLI 会自动删除关联的服务发现服务。

```
ecs-cli compose --project-name frontend service rm
```

```
INFO[0000] Updated ECS service successfully               desiredCount=0
 serviceName=frontend
INFO[0001] Service status                                 desiredCount=0 runningCount=1
 serviceName=frontend
INFO[0016] Service status                                 desiredCount=0 runningCount=0
 serviceName=frontend
INFO[0016] (service frontend) has stopped 1 running tasks: (task 824b5a76-8f9c-4beb-
a64b-6904e320630e).  timestamp="2018-09-12 00:37:25 +0000 UTC"
INFO[0016] ECS Service has reached a stable state         desiredCount=0 runningCount=0
 serviceName=frontend
INFO[0016] Deleted ECS service                            service=frontend
INFO[0016] ECS Service has reached a stable state         desiredCount=0 runningCount=0
 serviceName=frontend
INFO[0027] Waiting for your Service Discovery Service resource to be deleted...
INFO[0027] Cloudformation stack status                    stackStatus=DELETE_IN_PROGRESS
```

6. 要完成清除，请删除 `backend` 服务以及使用它创建的私有 DNS 命名空间。Amazon ECS CLI 会将私有 DNS 命名空间的 AWS CloudFormation 堆栈与为其创建该堆栈的 Amazon ECS 服务相关联。在删除服务时，也会删除命名空间。

```
ecs-cli compose --project-name backend service rm --delete-namespace
```

Amazon ECS 命令行参考

以下命令在 Amazon ECS CLI 中可用。通过向最终命令参数追加 `--help` 选项提供每条命令的帮助文本。使用以下命令列出 Amazon ECS CLI 的帮助文本：

```
ecs-cli --help
```

> **Note**
>
> 请确保您使用的是最新版本的 Amazon ECS CLI。最新版本为 1.15.1。有关发行说明，请参阅更改日志。

可用命令

- ecs-cli (p. 445)
- ecs-cli configure (p. 447)
- ecs-cli up (p. 453)
- ecs-cli down (p. 460)
- ecs-cli scale (p. 462)
- ecs-cli ps (p. 463)
- ecs-cli push (p. 465)
- ecs-cli pull (p. 467)
- ecs-cli images (p. 468)
- ecs-cli license (p. 471)
- ecs-cli compose (p. 472)
- ecs-cli compose service (p. 482)
- ecs-cli logs (p. 503)
- ecs-cli check-attributes (p. 505)
- ecs-cli registry-creds (p. 506)
- ecs-cli local (p. 511)
- 使用 Docker 复合文件语法 (p. 518)
- 使用 Amazon ECS 参数 (p. 520)

ecs-cli

描述

Amazon ECS 命令行界面 (CLI) 提供用于简化从本地开发环境创建、更新和监控集群和任务的高级命令。Amazon ECS; CLI 支持 Docker Compose。Docker Compose 是一个流行的开源工具，用于定义和运行多容器应用程序。

有关 Amazon ECS CLI 的快速演练，请参阅 教程：使用 Amazon ECS CLI 创建包含 Fargate 任务的集群 (p. 433)。

带 ecs-cli ***subcommand*** --help 的每个单独子命令都提供了帮助文本。

Important

所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli [--version] [*subcommand*] [--help]

选项

名称	描述
`--version, -v`	打印 Amazon ECS CLI 的版本信息。 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

可用子命令

ecs-cli 命令支持以下子命令：

configure

配置您的 AWS 凭证、要使用的区域以及要与 Amazon ECS CLI 结合使用的 ECS 集群名称。有关更多信息，请参阅ecs-cli configure (p. 447)。

migrate

将旧配置文件 (ECS CLI v0.6.6 及更早版本) 迁移到新的配置文件格式 (ECS CLI v1.0.0 及更高版本)。此命令将打印要进行的更改的摘要，然后请求确认以继续。有关更多信息，请参阅 ecs-cli configure migrate (p. 452)。

up

创建 ECS 集群（如果该集群不存在）和设置该集群所需的 AWS 资源。有关更多信息，请参阅ecs-cli up (p. 453)。

down

删除由 ecs-cli up 创建的 AWS CloudFormation 堆栈和关联的资源。有关更多信息，请参阅ecs-cli down (p. 460)。

scale

修改 ECS 集群中的容器实例数。有关更多信息，请参阅 ecs-cli scale (p. 462)。

logs

从 CloudWatch Logs 中检索容器日志。仅对使用 `awslogs` 驱动程序且指定了日志流前缀的任务有效。有关更多信息，请参阅 ecs-cli logs (p. 503)。

ps

列出一个 ECS 集群中运行的所有容器。有关更多信息，请参阅 ecs-cli ps (p. 463)。

push

将映像推送到 Amazon ECR 存储库。有关更多信息，请参阅 ecs-cli push (p. 465)。

pull

从 ECR 存储库中拉取映像。有关更多信息，请参阅 ecs-cli pull (p. 467)。

images

列出一个 ECS 集群中运行的所有容器。有关更多信息，请参阅ecs-cli images (p. 468)。

license

打印 Amazon ECS CLI 的 `LICENSE` 文件及其依赖项。有关更多信息，请参阅ecs-cli license (p. 471)。

compose

对 ECS 集群执行 docker-compose 式命令。有关更多信息，请参阅ecs-cli compose (p. 472)。

help

显示指定命令的帮助文本。

ecs-cli configure

配置要使用的 AWS 区域、资源创建前缀以及要与 Amazon ECS CLI 结合使用的 Amazon ECS 集群名称。将单一命名集群配置存储在 `~/.ecs/config` 文件中。创建的第一个集群配置将设置为默认配置。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

使用多个集群配置

使用多个集群配置时应注意以下事项：

- 可以存储多个集群配置，但始终有一个配置为默认配置。
- 存储的第一个集群配置将设置为默认配置。
- 使用 `ecs-cli configure default` 命令可更改要设置为默认配置的集群配置。有关更多信息，请参阅 ecs-cli configure default (p. 449)。
- 可使用 `--cluster-config` 标记在命令中引用非默认集群配置。

有关更多信息，请参阅 ecs-cli configure default (p. 449)。

> **Note**
>
> 确保您使用的是最新版本的 Amazon ECS CLI 才能使用所有配置选项。

语法

ecs-cli configure --cluster ***cluster_name*** --region ***region*** [--config-name ***config_name***] [--cfn-stack-name ***stack_name***] [--default-launch-type ***launch_type***] [--help]

选项

名称	描述
`--cluster, -c` *cluster_name*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串

名称	描述
	必需：是
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 ecs-cli configure 或 aws configure 命令配置的区域。 类型：字符串 必需：是
`--config-name` *`config_name`*	指定此集群配置的名称。这是可使用 `--cluster-config` 标记在命令中引用的名称。如果忽略此选项，此名称将设置为 `default`。 类型：字符串 必需：否
`--cfn-stack-name` *`stack_name`*	指定要添加到在 ecs-cli up 中创建的 AWS CloudFormation 堆栈的堆栈名称。 Important 建议不要使用此参数。包含此参数是为了确保与 ECS CLI 的早期版本的向后兼容性。 类型：字符串 默认值：`amazon-ecs-cli-setup-<cluster_name>` 必需：否
`--default-launch-type` *`launch_type`*	指定要使用的默认启动类型。有效值为 `FARGATE` 或 `EC2`。如果未指定，则不使用默认启动类型。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将 Amazon ECS CLI 配置为创建名为 `ecs-cli-demo` 的集群配置，该配置使用 `FARGATE` 作为集群 `ecs-cli-demo` 在 `us-east-1` 区域中的默认启动类型。

```
ecs-cli configure --region us-east-1 --cluster ecs-cli-demo --default-launch-type FARGATE
 --config-name ecs-cli-demo
```

输出：

```
INFO[0000] Saved ECS CLI cluster configuration ecs-cli-demo.
```

运行以下命令后的 `~/.ecs/config` 文件内容：

```
version: v1
default: ecs-cli-demo
clusters:
  ecs-cli-demo:
    cluster: ecs-cli-demo
    region: us-east-1
    default_launch_type: FARGATE
```

ecs-cli configure default

设置默认情况下要从中读取的集群配置。

> **Note**
>
> 与 AWS CLI 不同，Amazon ECS CLI 不期望或要求默认配置的名称为 `default`。配置名称不决定配置是否为默认配置。
>
> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli configure default --config-name ***config_name*** [--help]

选项

名称	描述
`--config-name` *`config_name`*	在后续命令中指定默认情况下要使用的集群配置的名称。 类型：字符串 必需：是
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将 Amazon ECS CLI 配置为将 `ecs-cli-demo` 集群配置设置为默认配置。

```
ecs-cli configure default --config-name ecs-cli-demo
```

如果此命令成功，则无任何输出。

ecs-cli configure profile

在指定的 Amazon ECS 配置文件（存储在 `~/.ecs/credentials` 文件中）中配置您的 AWS 凭证。如果创建多个配置文件，则可使用 ecs-cli configure profile default 命令更改默认情况下使用的配置文件。有关更多信息，请参阅ecs-cli configure profile default (p. 451)。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

您可以通过多种方式配置 AWS 凭证：

- 您可以设置 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 `AWS_SESSION_TOKEN` 环境变量。当您运行 ecs-cli configure profile 时，这些变量的值将存储在 Amazon ECS CLI 配置文件中。
- 您可以使用 `--access-key`、`--secret-key` 和 `--session-token` 选项直接在命令行上传递凭证。
- 您可以使用 `--profile-name` 标记提供新的配置文件名称。如果未提供配置文件名称，则配置文件名为 `default`。
- 配置的第一个配置文件将设置为默认配置文件。除非使用 `--ecs-profile` 标记，否则 Amazon ECS CLI 将使用此配置文件中指定的凭证。

使用多个配置文件

使用多个配置文件时应注意以下事项：

- 可以配置多个配置文件，但始终有一个配置文件为默认配置文件。在运行需要凭证的 Amazon ECS CLI 命令时使用此配置文件。
- 创建的第一个配置文件将设置为默认配置文件。
- 要更改默认配置文件，请使用 `ecs-cli configure profile default` 命令。有关更多信息，请参阅ecs-cli configure profile default (p. 451)。
- 可使用 `--ecs-profile` 标记在命令中引用非默认配置文件。

语法

ecs-cli configure profile --profile-name ***profile_name*** --access-key ***aws_access_key_id*** --secret-key ***aws_secret_access_key*** [--session-token ***token***] [--help]

选项

名称	描述
`--profile-name` *profile_name*	指定此 ECS 配置文件的名称。这是可使用 `--ecs-profile` 标记在命令中引用的名称。如果忽略此选项，此名称将设置为 `default`。 类型：字符串 必需：是
`--access-key` *aws_access_key_id*	指定要使用的 AWS 访问密钥。如果在运行 ecs-cli configure profile 时设置 `AWS_ACCESS_KEY_ID` 环境变量，则 AWS 访问密钥 ID 将设置为该环境变量的值。 类型：字符串 必需：是
`--secret-key` *aws_secret_access_key*	指定要使用的 AWS 私有密钥。如果在运行 ecs-cli configure profile 时设置 `AWS_SECRET_ACCESS_KEY` 环境变量，则 AWS 秘密访问密钥将设置为该环境变量的值。

名称	描述
	类型：字符串 必需：是
`--session-token` *`token`*	指定要使用的 AWS 会话令牌。如果在运行 ecs-cli configure profile 时设置 `AWS_SESSION_TOKEN` 环境变量，则 AWS 会话令牌将设置为该环境变量的值。有关使用会话令牌进行临时访问的更多信息，请参阅请求临时安全凭证。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例 1

此示例将 Amazon ECS CLI 配置为创建和使用名为 `default` 的配置文件与一组访问密钥。

```
ecs-cli configure profile --profile-name default --access-key $AWS_ACCESS_KEY_ID --secret-key $AWS_SECRET_ACCESS_KEY
```

输出：

```
INFO[0000] Saved ECS CLI profile configuration default.
```

示例 2

此示例将 Amazon ECS CLI 配置为创建和使用名为 `default` 的配置文件以及一组访问密钥和一个 AWS 会话令牌。

```
ecs-cli configure profile --profile-name default --access-key $AWS_ACCESS_KEY_ID --secret-key $AWS_SECRET_ACCESS_KEY --session-token $AWS_SESSION_TOKEN
```

输出：

```
INFO[0000] Saved ECS CLI profile configuration default.
```

ecs-cli configure profile default

设置默认情况下要从中读取的 Amazon ECS 配置文件。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli configure profile default --profile-name ***`profile_name`*** [--help]

选项

名称	描述
`--profile-name` *`profile_name`*	指定要标记为默认配置文件的 ECS 配置文件的名称。 类型：字符串 必需：是
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将 Amazon ECS CLI 配置为将 `default` 配置文件设置为要使用的默认配置文件。

```
ecs-cli configure profile default --profile-name default
```

如果此命令成功，则无任何输出。

ecs-cli configure migrate

将旧配置文件 (ECS CLI v0.6.6 及更早版本) 迁移到新的配置文件格式 (ECS CLI v1.0.0 及更高版本)。此命令将打印要进行的更改的摘要，然后请求确认以继续。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli configure migrate [--force] [--help]

选项

名称	描述
`--force`	忽略配置文件迁移过程中通常会出现的交互式描述和确认步骤。 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将旧 Amazon ECS CLI 配置文件迁移到新的 YAML 格式。

```
ecs-cli configure migrate
```

ecs-cli up

创建 Amazon ECS 集群（如果该集群不存在）和设置该集群所需的 AWS 资源。

此命令创建名为 `amazon-ecs-cli-setup-cluster_name` 的新 AWS CloudFormation 堆栈。您可以在 AWS 管理控制台中查看堆栈创建的进度。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

```
ecs-cli up [--capability-iam | --instance-role instance-
profile-name] [--keypair keypair_name] [--size n] [--azs
availability_zone_1,availability_zone_2] [--security-group
security_group_id[,security_group_id[,...]]] [--cidr ip_range] [--port
port_number] [--subnets subnet_1,subnet_2] [--vpc vpc_id] [--extra-user-
data string] [--instance-type instance_type] [--image-id ami_id] [--launch-
type launch_type] [--no-associate-public-ip-address] [--force] [--tags
key1=value1,key2=value2] [--cluster cluster_name] [--region region] [--empty]
[--verbose] [--help]
```

选项

名称	描述
`--capability-iam`	确认此命令可创建 IAM 资源。 **Note** 此参数仅对于使用 EC2 启动类型的任务受支持。 如果您未使用 `--instance-role` 指定实例配置文件名称，则此参数是必需的。您无法同时指定两个选项。 必需：否
`--keypair keypair_name`	指定用于支持对集群中的 EC2 实例进行 SSH 访问的现有 Amazon EC2 密钥对的名称。 **Note** 此参数仅对于使用 EC2 启动类型的任务受支持。 有关创建密钥对的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 使用 Amazon EC2 进行设置。 类型：字符串 必需：否
`--size n`	指定要启动并注册到集群的实例的数量。

名称	描述
	Note 此参数仅对于使用 EC2 启动类型的任务受支持。 类型：整数 默认值：1 必需：否
`--azs` *`availability_zone_1,availability_zone_2`*	指定要在其中创建子网的两个 VPC 可用区的逗号分隔的列表(这两个区域必须具有 `available` 状态)。如果您未使用 `--vpc` 选项指定 VPC ID，我们建议使用此选项。 Warning 将此选项保留为空可能导致在随机选择的区域不可用时无法启动容器实例。 类型：字符串 必需：否
`--security-group` *`security_group_id`*`[,`*`security_group_id`*`]`	指定要与容器实例关联的现有安全组的逗号分隔列表。如果您在此处未指定安全组，请创建一个新的安全组。 有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Security Groups 部分。 必需：否
`--cidr` *`ip_range`*	指定要用于集群中的容器实例的安全组的 CIDR/IP 范围。 Note 如果已使用 `--security-group` 选项指定现有安全组，则忽略此参数。 类型：CIDR/IP 范围 默认值：`0.0.0.0/0` 必需：否
`--port` *`port_number`*	指定要在安全组上打开以用于集群中的容器实例的端口。 Note 如果已使用 `--security-group` 选项指定现有安全组，则忽略此参数。 类型：整数 默认值：`80` 必需：否

名称	描述
`--subnets` *`subnet_1,subnet_2`*	指定要在其中启动容器实例的现有 VPC 子网 ID 的逗号分隔的列表。 类型：字符串 必需：如果已使用 `--vpc` 选项指定 VPC，则需要此选项。
`--vpc` *`vpc_id`*	指定要在其中启动容器实例的现有 VPC 的 ID。如果指定了一个 VPC ID，则必须使用 `--subnets` 选项指定该 VPC 中的现有子网的列表。如果未指定 VPC ID，请使用两个子网创建一个新的 VPC。 类型：字符串 必需：否
`--extra-user-data` *`string`*	指定容器实例的其他用户数据。文件可以是 shell 脚本或 cloud-init 指令。它们与 Amazon ECS CLI 提供的用于指示实例加入您的集群的用户数据一起打包到 MIME 分段存档中。有关更多信息，请参阅指定用户数据 (p. 458)。 类型：字符串 必需：否
`--instance-type` *`instance_type`*	指定您的容器实例的 Amazon EC2 实例类型。如果您指定 A1 实例类型，例如 `a1.medium`，并省略 `--image-id` 参数，ECS CLI 将使用该容器实例的 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI AMI ID。 Note 仅支持对使用 EC2 启动类型的任务使用此参数。 有关 EC2 实例类型的更多信息，请参阅 Amazon EC2 实例。 类型：字符串 默认值：`t2.micro` 必需：否

名称	描述
`--image-id` *`ami_id`*	指定要用于容器实例的 Amazon EC2 AMI ID。 如果您未指定 AMI ID，Amazon ECS CLI 将自动检索最新稳定的 Amazon ECS-optimized Amazon Linux 2 AMI（方式为在创建集群资源过程中查询 Systems Manager Parameter Store API）。这要求您正在使用的用户账户具有所需的 Systems Manager 权限。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。 如果您为 `--instance-type` 参数指定 A1 实例类型并忽略 `--image-id` 参数，ECS CLI 将使用该容器实例的 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI AMI ID。 Note 仅支持对使用 EC2 启动类型的任务使用此参数。 类型：字符串 默认值：指定区域经 Amazon ECS 优化的最新稳定的 AMI。 必需：否
`--no-associate-public-ip-address`	不向该 VPC 中的新实例分配公有 IP 地址。除非指定此选项，否则该 VPC 中的新实例会收到自动分配的公有 IP 地址。 Note 此参数仅对于使用 EC2 启动类型的任务受支持。 必需：否
`--force, -f`	强制重新创建与您的当前配置匹配的所有现有资源。此选项对于从先前失败的尝试中清除过期资源非常有用。 必需：否
`--tags` *`key1=value1,key2=value2`*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签采用以下格式：`key1=value1,key2=value2,key3=value3`。有关更多信息，请参阅 标记资源 (p. 458)。 类型：键值对 必需：否
`--instance-role, -f` *`instance-profile-name`*	为您的集群中的实例指定自定义 IAM 实例配置文件名称。 Note 此参数仅对于使用 EC2 启动类型的任务受支持。 如果您未指定 `--capability-iam` 选项，则此参数是必需的。您无法同时指定两个选项。 必需：否

名称	描述
`--launch-type` *`launch_type`*	指定要使用的启动类型。可用的选项有 `FARGATE` 或 `EC2`。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。 这将覆盖在集群配置中存储的默认启动类型。 类型：字符串 必需：否
`--verbose, --debug`	打开调试日志记录。这将提供帮助诊断问题的更详细的命令输出。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--empty, -e`	指定将创建一个没有任何资源的 ECS 集群。如果还指定了其他将创建资源的标志，将忽略它们并显示警告。 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

指定用户数据

在启动使用 EC2 启动类型的任务时，ECS CLI 始终创建包含以下用户数据的容器实例：

```
#!/bin/bash
echo ECS_CLUSTER={ clusterName } >> /etc/ecs/ecs.config
```

此用户数据指示容器实例加入您的 ECS 集群。您可以选择使用 `--extra-user-data` 标志包括其他用户数据。可以多次指定该标志。例如，额外的用户数据可以是 Shell 脚本或 cloud-init 指令。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的启动时在 Linux 实例上运行命令。

Amazon ECS CLI 获取用户数据并将其打包到 MIME 分段存档中，这可供容器实例上的 cloud-init 使用。Amazon ECS CLI 允许使用 `--extra-user-data` 传入现有 MIME 分段存档。Amazon ECS CLI 会解包现有存档，然后将其重新打包为最终存档（保留所有标头和内容类型信息）。以下是示例：

```
ecs-cli up \
  --capability-iam \
  --extra-user-data my-shellscript \
  --extra-user-data my-cloud-boot-hook \
  --extra-user-data my-mime-multipart-archive \
  --launch-type EC2
```

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

如果您在使用 ecs-cli up 命令时指定资源标签，则可以标记 Amazon ECS 集群以及由 AWS CloudFormation 堆栈创建的以下资源：

- 容器实例

  > **Note**
  >
  > 为了让您的容器实例允许标签，您需要选择使用新的 Amazon ECS 资源 ARN 格式。有关更多信息，请参阅 Amazon 资源名称 (ARN) 和 ID (p. 121)。

- VPC
- 子网
- Internet 网关
- 路由表
- 安全组
- Autoscaling 组

> **Note**
>
> 对于 Autoscaling 组，ECS CLI 将添加其值为 `ECS Instance - <CloudFormation stack name>` 的 `Name` 标签，该标签将传播到您的容器实例。您可以通过指定自己的 `Name` 标签来覆盖此行为。

示例

为使用 EC2 启动类型的任务创建集群

此示例将启动一个包含 4 个 `c4.large` 容器实例的集群并将它们配置为使用名为 `id_rsa` 的 EC2 密钥对。

```
ecs-cli up --keypair id_rsa --capability-iam --size 4 --instance-type c4.large --launch-
type EC2
```

输出：

```
INFO[0001] Using recommended Amazon Linux AMI with ECS Agent 1.17.3 and Docker version
 17.12.1-ce
INFO[0000] Created cluster                                  cluster=ecs-cli-ec2-demo
INFO[0000] Waiting for your cluster resources to be created
INFO[0001] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0061] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0121] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0181] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
Cluster creation succeeded.
VPC created: vpc-abcd1234
Security Group created: sg-abcd1234
Subnets created: subnet-abcd1234
Subnets created: subnet-dcba4321
```

利用使用 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI 的容器实例创建集群

此示例将启动一个包含一个 `a1.medium` 容器实例（该实例将使用 Amazon ECS-optimized Amazon Linux 2 (arm64) AMI）的集群。

```
ecs-cli up --capability-iam --instance-type a1.medium --launch-type EC2 --region us-east-2
```

输出：

```
WARN[0000] You will not be able to SSH into your EC2 instances without a key pair.
INFO[0000] Using Arm ecs-optimized AMI because instance type was a1.medium
INFO[0001] Using recommended Amazon Linux 2 AMI with ECS Agent 1.25.3 and Docker version
 18.06.1-ce
INFO[0000] Created cluster                                  cluster=ecs-cli-ec2-demo
INFO[0000] Waiting for your cluster resources to be created
INFO[0001] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0061] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0121] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
INFO[0181] Cloudformation stack status                      stackStatus=CREATE_IN_PROGRESS
Cluster creation succeeded.
VPC created: vpc-abcd1234
Security Group created: sg-abcd1234
Subnets created: subnet-abcd1234
Subnets created: subnet-dcba4321
```

为使用 Fargate 启动类型的任务创建集群

此示例将为您的 Fargate 任务创建一个集群并创建带两个子网的新 VPC。

```
ecs-cli up --launch-type FARGATE
```

输出：

```
INFO[0001] Created cluster                                  cluster=ecs-cli-fargate-demo
 region=us-west-2
INFO[0003] Waiting for your cluster resources to be created...
```

```
INFO[0003] Cloudformation stack status                   stackStatus="CREATE_IN_PROGRESS"
INFO[0066] Waiting for your cluster resources to be created...
INFO[0066] Cloudformation stack status                   stackStatus="CREATE_IN_PROGRESS"
VPC created: vpc-abcd1234
Subnets created: subnet-abcd1234
Subnets created: subnet-dcba4321
Cluster creation succeeded.
```

创建一个空集群

此示例将创建一个名为 `ecs-cli-empty-demo` 的没有任何资源的空集群。

```
ecs-cli up --empty --cluster ecs-cli-empty-demo
```

输出:

```
INFO[0000] Created cluster                               cluster=ecs-cli-empty-demo
 region=us-east-1
Cluster creation succeeded.
```

ecs-cli down

删除由 ecs-cli up 创建的 AWS CloudFormation 堆栈和关联的资源。

> **Note**
>
> Amazon ECS CLI 只能管理使用 CLI 创建的任务、服务和容器实例。要管理并非使用 Amazon ECS CLI 创建的任务、服务和容器实例，请使用 AWS Command Line Interface 或 AWS 管理控制台。

ecs-cli down 命令尝试删除 `~/.ecs/config` 中指定的集群。但是，如果您的集群中有任何活动服务（即便预期数量为 0）或不是由 ecs-cli up 创建的已注册的容器实例，则集群不会被删除，并且服务和预先存在的容器实例将保持活动状态。例如，如果您使用带已注册的容器实例的现有 ECS 集群 (如默认集群)，则可能发生这种情况。

如果您的集群中有要删除的剩余服务或容器实例，可执行清除 Amazon ECS 资源 (p. 26)中的过程来删除它们，然后删除集群。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

```
ecs-cli down [--force] [--cluster cluster_name] [--region region] [--help]
```

选项

名称	描述
`--force, -f`	确认此命令将永久删除资源并绕过确认提示。 必需：否
`--region, -r region`	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。

名称	描述
	类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例 1

此示例将删除包含资源的集群。

```
ecs-cli down --cluster ecs-cli-fargate-demo --force
```

输出：

```
INFO[0001] Waiting for your cluster resources to be deleted
INFO[0001] Cloudformation stack status                   stackStatus=DELETE_IN_PROGRESS
INFO[0062] Cloudformation stack status                   stackStatus=DELETE_IN_PROGRESS
INFO[0123] Cloudformation stack status                   stackStatus=DELETE_IN_PROGRESS
INFO[0154] Deleted cluster
```

示例 2

此示例将删除空集群。

```
ecs-cli down --cluster ecs-cli-empty-demo --force
```

输出:

```
INFO[0002] No CloudFormation stack found for cluster 'ecs-cli-empty-demo'.
INFO[0003] Deleted cluster                                  cluster=ecs-cli-empty-demo
```

ecs-cli scale

修改集群中的容器实例的数量。此命令更改 Auto Scaling 组中由 ecs-cli up 命令创建的所需实例计数和最大实例计数。您可以使用此命令扩展 (增加实例数) 或缩减 (减少实例数) 您的集群。

> **Note**
>
> Amazon ECS CLI 只能管理使用 CLI 创建的任务、服务和容器实例。要管理并非使用 Amazon ECS CLI 创建的任务、服务和容器实例，请使用 AWS Command Line Interface 或 AWS 管理控制台。
>
> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli scale --capability-iam --size *n* [--cluster *cluster_name*] [--region *region*] [--help]

选项

名称	描述
`--capability-iam`	确认此命令可创建 IAM 资源。 必需：是
`--size` *n*	指定要在集群中保留的实例的数量。 类型：整数 必需：是
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。

名称	描述
	类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将当前集群扩展至两个容器实例。

```
ecs-cli scale --size 2 --capability-iam
```

输出:

```
INFO[0001] Waiting for your cluster resources to be updated
INFO[0001] Cloudformation stack status                   stackStatus=UPDATE_IN_PROGRESS
```

ecs-cli ps

列出 Amazon ECS 集群中运行的所有容器。

Amazon ECS CLI 显示的 IP 地址主要取决于您配置任务和集群的方式：

- 对于使用 EC2 启动类型而没有任务联网的任务，显示的 IP 地址是运行您的任务的 Amazon EC2 实例的公有 IP 地址，或实例私有 IP 地址（如果它缺少公有 IP 地址）。
- 对于使用 EC2 启动类型且有任务联网的任务，ECS CLI 将仅显示从任务的“描述任务”输出的网络接口部分中获取的私有 IP 地址。
- 对于使用 Fargate 启动类型的任务，Amazon ECS CLI 将返回为附加到 Fargate 任务的弹性网络实例分配的公有 IP 地址。如果弹性网络实例缺少公有 IP 地址，则 Amazon ECS CLI 将回退到从“描述任务”输出的网络接口部分中获取的私有 IP 地址。

语法

`ecs-cli ps [--desired-status` *`status`*`] [--cluster` *`cluster_name`*`] [--region` *`region`*`] [--help]`

选项

名称	描述
`--desired-status` *`status`*	筛选容器列表结果时使用的容器所需状态。 必需：否 有效值：`RUNNING` | `STOPPED`
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例显示正在集群中运行的容器。

```
ecs-cli ps
```

输出:

```
Name                                                State    Ports
 TaskDefinition       Health
afd7f8a0-3813-4e1a-9d9e-ca7e9d1fcfbb/wordpress   RUNNING  36.253.177.221:80->80/tcp
 compose3:7           HEALTHY
dca67e02-68ca-4507-b194-a47239b5e7a9/wordpress   RUNNING  37.234.146.14:80->80/tcp
 healthcheck:3        UNKNOWN
dca67e02-68ca-4507-b194-a47239b5e7a9/redis       RUNNING
 healthcheck:3        HEALTHY
feb6e10e-3385-4c9b-a6cb-787cc8e90dda/sample-app  RUNNING  54.229.211.206:80->80/tcp
 tutorial-task-def:1  UNKNOWN
```

ecs-cli push

将映像推送到 Amazon ECR 存储库。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli push [--registry-id *registry_id*] [--tags *key1=value1,key2=value2*] [--region *region*] [--verbose] [--use-fips] *ECR_REPOSITORY*[:*TAG*] [--help]

选项

名称	描述
`--registry-id` *registry_id*	指定要将映像推送到的 Amazon ECR 注册表 ID。默认情况下，映像将推送到当前 AWS 账户。 类型：字符串 必需：否
`--tags` *value*	指定要应用于 Amazon ECR 存储库的元数据。每一个标签都包含一个键和一个可选值。标签键的最大字符长度为 128 个字符，标签值的最大长度为 256 个字符。标签采用以下格式：`key1=value1,key2=value2,key3=value3`。 类型：键值对 必需：否
`--verbose, --debug`	打开调试日志记录。这将提供帮助诊断问题的更详细的命令输出。 必需：否
`--use-fips`	通过 FIPS 终端节点将调用路由到 Amazon ECR。 必需：否
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串

名称	描述
	必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用 FIPS 终端节点

Amazon ECS CLI 支持使用 FIPS 终端节点调用 Amazon ECR。为确保您使用的是 FIPS 终端节点访问 Amazon ECR，请对 push、pull 或 images 命令使用 `--use-fips` 标记。FIPS 终端节点目前仅在 us-west-1、us-west-2、us-east-1、us-east-2 和 AWS GovCloud (US) 中可用。有关更多信息，请参阅美国联邦信息处理标准 (FIPS) 140-2。

示例

示例 1

此示例将名为 `ubuntu` 的本地映像推送到具有相同名称的 Amazon ECR 存储库。

```
ecs-cli push ubuntu
```

输出:

```
INFO[0000] Getting AWS account ID...
INFO[0000] Tagging image
 repository="aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu" source-image=ubuntu
 tag=
INFO[0000] Image tagged
```

```
INFO[0001] Creating repository                          repository=ubuntu
INFO[0001] Repository created
INFO[0001] Pushing image
 repository="aws_account_id.dkr.ecr.us-east-1.amazonaws.com/ubuntu" tag=
INFO[0079] Image pushed
```

ecs-cli pull

从 Amazon ECR 存储库中拉取映像。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli pull [--registry-id *registry_id*] [--region *region*] [--verbose] [--use-fips] *ECR_REPOSITORY*[:*TAG*|@*DIGEST*] [--help]

选项

名称	描述
`--registry-id` *registry_id*	指定要从中拉取映像的 Amazon ECR 注册表 ID。默认情况下，将从当前 AWS 账户中拉取映像。 必需：否
`--verbose, --debug`	打开调试日志记录。这将提供帮助诊断问题的更详细的命令输出。 必需：否
`--use-fips`	通过 FIPS 终端节点将调用路由到 Amazon ECR。 必需：否
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否

名称	描述
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用 FIPS 终端节点

Amazon ECS CLI 支持使用 FIPS 终端节点调用 Amazon ECR。为确保您使用的是 FIPS 终端节点访问 Amazon ECR，请对 push、pull 或 images 命令使用 `--use-fips` 标记。FIPS 终端节点目前仅在 us-west-1、us-west-2、us-east-1、us-east-2 和 AWS GovCloud (US) 中可用。有关更多信息，请参阅美国联邦信息处理标准 (FIPS) 140-2。

示例

示例 1

此示例从具有相同名称的 Amazon ECR 存储库中拉取名为 `amazonlinux` 的本地映像。

```
ecs-cli pull amazonlinux
```

输出:

```
INFO[0000] Getting AWS account ID...
INFO[0000] Pulling image
 repository="aws_account_id.dkr.ecr.us-east-1.amazonaws.com/amazonlinux" tag=
INFO[0129] Image pulled
```

ecs-cli images

列出 Amazon ECR 注册表或存储库中的映像。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

```
ecs-cli images [--registry-id registry_id] [--tagged|--untagged] [--region
region] [--verbose] [--use-fips] [ECR_REPOSITORY] [--help]
```

选项

名称	描述
`--registry-id` *registry_id*	指定具有要列出映像的 Amazon ECR 注册表。默认情况下，将列出当前 AWS 账户的映像。 必需：否
`--tagged`	筛选结果以仅显示标记的映像。 必需：否
`--untagged`	筛选结果以仅显示未标记的映像。 必需：否
`--verbose, --debug`	打开调试日志记录。这将提供帮助诊断问题的更详细的命令输出。 必需：否
`--use-fips`	通过 FIPS 终端节点将调用路由到 Amazon ECR。 必需：否
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *aws_profile*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *cluster_name*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否

名称	描述
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用 FIPS 终端节点

Amazon ECS CLI 支持使用 FIPS 终端节点调用 Amazon ECR。为确保您使用的是 FIPS 终端节点访问 Amazon ECR，请对 push、pull 或 images 命令使用 `--use-fips` 标记。FIPS 终端节点目前仅在 us-west-1、us-west-2、us-east-1、us-east-2 和 AWS GovCloud 中可用。有关更多信息，请参阅美国联邦信息处理标准 (FIPS) 140-2。

示例

示例 1

此示例列出一个 Amazon ECR 注册表中的所有映像。

```
ecs-cli images
```

输出：

```
REPOSITORY NAME          TAG                 IMAGE DIGEST
                            PUSHED AT           SIZE
rkt                      latest
 sha256:404758ad8af94347fc8582fc8e30b6284f2b0751de29b2e755da212f80232fac   3 months ago
    203 MB
foobuntu                 latest
 sha256:6b079ae764a6affcb632231349d4a5e1b084bece8c46883c099863ee2aeb5cf8   4 days ago
    51.7 MB
ubuntu                   xenial
 sha256:6b079ae764a6affcb632231349d4a5e1b084bece8c46883c099863ee2aeb5cf8   4 days ago
    51.7 MB
ubuntu                   latest
 sha256:6b079ae764a6affcb632231349d4a5e1b084bece8c46883c099863ee2aeb5cf8   4 days ago
    51.7 MB
ubuntu                   <none>
 sha256:512e30a26d9fa3648dbccb9e78e9bab636e6022e2d80bd73c99177b21a0d3982   19 minutes ago
    268 MB
ubuntu                   trusty
 sha256:bd6d24e8fa3f5822146b2c94247976b87e6564195c3c180b67833e6ea699f7c2   18 minutes ago
    67.2 MB
ubuntu                   precise
 sha256:b38267a51fb4460699bc2bcdbb53d42fec697bb4e4f9a819df3e762cec393b2a   17 minutes ago
    40.1 MB
amazon-ecs-sample        latest
 sha256:bf04071a8edecc309f4d109ae36f24a5c272a115b6f7e636f77940059024d71c   2 weeks ago
    105 MB
golang                   latest
 sha256:137b22efee2df470b0cd28ebfc1ae583be0baf09334a5a882096193577d983ab   4 days ago
    266 MB
amazonlinux              latest
 sha256:a59d563b5139deee8cb108bfb97bf3e9021b8ccea6dec8ff49733230cb2f0eca   4 days ago
    98.8 MB
awsbatch/fetch_and_run   latest
 sha256:543800007416d0ccff4f63643bb18eeff4b874ea772128efcdc231ff456a37fc   6 weeks ago
    116 MB
```

示例 2

此示例列出一个特定 Amazon ECR 存储库中的所有映像。

```
ecs-cli images ubuntu
```

输出：

```
REPOSITORY NAME     TAG                 IMAGE DIGEST
                       PUSHED AT           SIZE
ubuntu              xenial
 sha256:6b079ae764a6affcb632231349d4a5e1b084bece8c46883c099863ee2aeb5cf8   4 days ago
    51.7 MB
ubuntu              latest
 sha256:6b079ae764a6affcb632231349d4a5e1b084bece8c46883c099863ee2aeb5cf8   4 days ago
    51.7 MB
ubuntu              <none>
 sha256:512e30a26d9fa3648dbccb9e78e9bab636e6022e2d80bd73c99177b21a0d3982   20 minutes ago
    268 MB
ubuntu              trusty
 sha256:bd6d24e8fa3f5822146b2c94247976b87e6564195c3c180b67833e6ea699f7c2   19 minutes ago
    67.2 MB
ubuntu              precise
 sha256:b38267a51fb4460699bc2bcdbb53d42fec697bb4e4f9a819df3e762cec393b2a   18 minutes ago
    40.1 MB
```

示例 3

此示例列出一个 Amazon ECR 注册表中所有未标记的映像。

```
ecs-cli image --untagged
```

输出:

```
REPOSITORY NAME     TAG                 IMAGE DIGEST
                       PUSHED AT           SIZE
ubuntu              <none>
 sha256:512e30a26d9fa3648dbccb9e78e9bab636e6022e2d80bd73c99177b21a0d3982   24 minutes ago
    268 MB
```

ecs-cli license

打印 Amazon ECS CLI 的 `LICENSE` 文件及其依赖项。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

`ecs-cli license [--help]`

选项

名称	描述
`--help, -h`	显示指定命令的帮助文本。

名称	描述
	必需：否

示例

示例

此示例打印许可证文件。

```
ecs-cli license
```

输出:

```
Copyright 2015 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License"). You may not use this file
 except in compliance with the
License. A copy of the License is located at

    http://aws.amazon.com/apache2.0/

or in the "license" file accompanying this file. This file is distributed on an "AS IS"
 BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the specific
 language governing permissions
and limitations under the License.
...
```

ecs-cli compose

在 ECS 集群上利用 docker-compose 式命令管理 Amazon ECS 任务。有关 Docker 复合文件语法如何与 Amazon ECS CLI 结合使用的更多信息，请参阅使用 Docker 复合文件语法 (p. 518)。

> **Note**
>
> 要使用 Amazon ECS CLI 创建 Amazon ECS 服务，请参阅 ecs-cli compose service (p. 482)。

ecs-cli compose 命令将项目名称与其创建的任务定义和服务结合使用。当 CLI 从复合文件创建任务定义时，任务定义将称为 *project-name*。当 CLI 从复合文件创建服务时，服务将称为 `service-`*project-name*。默认情况下，项目名称是包含 Docker 复合文件的目录的名称。但是，您也可以使用 `--project-name` 选项指定自己的项目名称。

> **Note**
>
> Amazon ECS CLI 只能管理使用 CLI 创建的任务、服务和容器实例。要管理并非使用 Amazon ECS CLI 创建的任务、服务和容器实例，请使用 AWS Command Line Interface 或 AWS 管理控制台。
>
> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli compose [--verbose] [--file ***compose_file***] [--project-name ***project_name***] [--task-role-arn ***task_role_arn***] [--ecs-params ***ecs_params_file***] [--registry-creds ***value***] [--region ***region***] [--cluster-config ***cluster_config_name***] [--ecs-profile ***ecs_profile***] [--aws-profile ***aws_profile***] [--cluster ***cluster_name***] [--help] [***subcommand***] [***arguments***] [--help]

选项

名称	描述
`--verbose, --debug`	提高命令输出的详细程度以帮助诊断。 必需：否
`--file, -f` *`compose-file`*	指定要使用的 Docker 复合文件。此时，最新版本的 Amazon ECS CLI 支持 Docker 复合文件语法版本 1 和 2。 如果在 ecs-cli compose 运行时设置 `COMPOSE_FILE` 环境变量，则 Docker 复合文件将设置为该环境变量的值。 类型：字符串 默认值：`./docker-compose.yml` 必需：否
`--project-name, -p` *`project-name`*	指定要使用的项目名称。如果在运行 ecs-cli compose 时设置 `COMPOSE_PROJECT_NAME` 环境变量，则项目名称将设置为该环境变量的值。 类型：字符串 默认值：当前目录名。 必需：否
`--task-role-arn` *`role_value`*	指定该任务中的容器可以代入的 IAM 角色的短名称或完整的 Amazon 资源名称 (ARN)。该任务中的所有容器都将被授予在该角色中指定的权限。 类型：字符串 必需：否
`--ecs-params`	指定非 Docker 复合文件固有的 ECS 参数。有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。 默认值：`./ecs-params.yml` 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。

名称	描述
	类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

可用子命令

ecs-cli compose 命令支持以下子命令。其中，每条子命令均具有关联的标记，这些标记可通过 `--help` 标记显示。

create

从复合文件创建 Amazon ECS 任务定义。有关更多信息，请参阅 ecs-cli compose create (p. 475)。

ps, list

列出集群中由复合项目启动的所有容器。

run [***`containerName`***] ["***`command`*** ..."] ...

使用为容器提供的一次性命令来启动覆盖命令的所有容器。

scale ***`n`***

将正在运行的任务数扩展至指定计数。

start

从通过复合文件创建的任务定义启动单个任务。有关更多信息，请参阅 ecs-cli compose start (p. 478)。

stop, down

停止由复合项目创建的所有正在运行的任务。

up

从复合文件创建 ECS 任务定义（如果该任务定义不存在）并在集群上运行该任务的一个实例（create 和 start 的组合）。有关更多信息，请参阅ecs-cli compose up (p. 480)。

service [***`subcommand`***]

从复合文件创建 ECS 服务。有关更多信息，请参阅 ecs-cli compose service (p. 482)。

help

显示指定命令的帮助文本。

ecs-cli compose create

从复合文件创建 Amazon ECS 任务定义。

> **Important**
>
> 建议不要对敏感信息（如凭证数据）使用纯文本环境变量。
>
> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli compose create [--region ***region***] [--cluster-config ***cluster_config_name***] [--ecs-profile ***ecs_profile***] [--aws-profile ***aws_profile***] [--cluster ***cluster_name***] [--launch-type ***launch_type***] [--create-log-groups] [--tags ***key1=value1,key2=value2***] [--help]

选项

名称	描述
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *aws_profile*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *cluster_name*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--launch-type` *launch_type*	指定要使用的启动类型。可用的选项有 `FARGATE` 或 `EC2`。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。

名称	描述
	这将覆盖在集群配置中存储的默认启动类型。 类型：字符串 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--tags` *`key1=value1,key2=value2`*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签采用以下格式：`key1=value1,key2=value2,key3=value3`。有关更多信息，请参阅 标记资源 (p. 476)。 类型：键值对 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

使用 `ecs-cli compose create` 命令时，使用 `--tags` 标记让您可以将元数据标签添加到任务定义。

示例

注册任务定义

此示例从 `hello-world` 复合文件创建项目名称为 `hello-world.yml` 的任务定义。

```
ecs-cli compose --project-name hello-world --file hello-world.yml create --launch-type EC2
```

输出：

```
INFO[0000] Using ECS task definition                     TaskDefinition=ecscompose-hello-
world:5
```

在没有任务联网的情况下使用 EC2 启动类型注册任务定义

此示例从 `hello-world` 复合文件创建项目名称为 `hello-world.yml` 的任务定义。为容器大小参数指定的其他 ECS 参数。

名为 `hello-world.yml` 的示例 Docker 复合文件：

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    logging:
```

```
      driver: awslogs
      options:
        awslogs-group: /ecs/cli/tutorial
        awslogs-region: us-east-1
        awslogs-stream-prefix: nginx
```

名为 `ecs-params.yml` 的示例 ECS 参数文件：

```
version: 1
task_definition:
  services:
    nginx:
      cpu_shares: 256
      mem_limit: 0.5GB
      mem_reservation: 0.5GB
```

```
ecs-cli compose --project-name hello-world --file hello-world.yml --ecs-params ecs-
params.yml --region us-east-1 create --launch-type EC2
```

输出：

```
INFO[0000] Using ECS task definition                     TaskDefinition=ecscompose-hello-
world:5
```

通过使用 Fargate 启动类型来注册任务定义

此示例从 `hello-world` 复合文件创建项目名称为 `hello-world.yml` 的任务定义。将为 Fargate 启动类型的任务网络配置指定其他 ECS 参数。然后，将运行任务的一个实例。

名为 `hello-world.yml` 的示例 Docker 复合文件：

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial
        awslogs-region: us-east-1
        awslogs-stream-prefix: nginx
```

名为 `ecs-params.yml` 的示例 ECS 参数文件：

```
version: 1
task_definition:
  task_execution_role: ecsTaskExecutionRole
  ecs_network_mode: awsvpc
  task_size:
    mem_limit: 0.5GB
    cpu_limit: 256
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - subnet-abcd1234
        - subnet-dbca4321
      security_groups:
```

```
        - sg-abcd1234
      assign_public_ip: ENABLED
```

命令:

```
ecs-cli compose --project-name hello-world --file hello-world.yml --ecs-params ecs-
params.yml --region us-east-1 create --launch-type FARGATE
```

输出：

```
INFO[0000] Using ECS task definition                    TaskDefinition=ecscompose-hello-
world:5
```

ecs-cli compose start

从通过复合文件创建的任务定义启动单个 Amazon ECS 任务。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli compose start [--region ***region***] [--cluster-config ***cluster_config_name***] [--ecs-profile ***ecs_profile***] [--aws-profile ***aws_profile***] [--cluster ***cluster_name***] [--launch-type ***launch_type***] [--create-log-groups] [--help]

选项

名称	描述
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *aws_profile*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否

名称	描述
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--launch-type` *`launch_type`*	指定要使用的启动类型。可用的选项有 `FARGATE` 或 `EC2`。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。 这将覆盖在集群配置中存储的默认启动类型。 类型：字符串 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

运行任务

此示例从 `hello-world.yml` 复合文件创建一个任务定义。将为 Fargate 启动类型的任务网络配置指定其他 ECS 参数。然后，将会使用该任务定义运行单个任务。

名为 `hello-world.yml` 的示例 Docker 复合文件：

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial
        awslogs-region: us-east-1
        awslogs-stream-prefix: nginx
```

名为 `ecs-params.yml` 的示例 ECS 参数文件：

```
version: 1
task_definition:
  task_execution_role: ecsTaskExecutionRole
  ecs_network_mode: awsvpc
  task_size:
    mem_limit: 0.5GB
    cpu_limit: 256
run_params:
  network_configuration:
```

```
    awsvpc_configuration:
      subnets:
        - subnet-abcd1234
        - subnet-dbca4321
      security_groups:
        - sg-abcd1234
      assign_public_ip: ENABLED
```

命令:

```
ecs-cli compose --file hello-world.yml --ecs-params ecs-params.yml start --launch-
type FARGATE --create-log-groups
```

输出：

```
INFO[0000] Using ECS task definition                     TaskDefinition=ecscompose-hello-
world:5
```

ecs-cli compose up

如果 Amazon ECS 任务定义不存在，则从您的复合文件创建一个定义并在您的集群上运行该任务的一个实例。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli compose up [--region *region*] [--cluster-config *cluster_config_name*] [--ecs-profile *ecs_profile*] [--aws-profile *aws_profile*] [--cluster *cluster_name*] [--launch-type *launch_type*] [--create-log-groups] [--force-update] [--tags *key1=value1,key2=value2*] [--disable-ecs-managed-tags] [--help]

选项

名称	描述
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否

名称	描述
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--launch-type` *`launch_type`*	指定要使用的启动类型。可用的选项有 `FARGATE` 或 `EC2`。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。 这将覆盖在集群配置中存储的默认启动类型。 类型：字符串 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--force-update`	强制任务重新启动。 必需：否
`--tags` *`key1=value1,key2=value2`*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签采用以下格式：`key1=value1,key2=value2,key3=value3`。有关更多信息，请参阅 标记资源 (p. 481)。 类型：键值对 必需：否
`--disable-ecs-managed-tags`	禁用 Amazon ECS 托管标签。有关更多信息，请参阅 标记资源以便于计费 (p. 332)。 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

使用 `ecs-cli compose up` 命令时，使用 `--tags` 标记使您能够将元数据标签添加到任务定义和任务。Amazon ECS 托管标签默认情况下将启用，除非使用 `--disable-ecs-managed-tags` 标记明确禁用它们。有关更多信息，请参阅标记资源以便于计费 (p. 332)。

示例

通过将 AWS Fargate 启动类型与任务联网结合使用来注册任务定义

此示例从 `hello-world` 复合文件创建项目名称为 `hello-world.yml` 的任务定义。将为 Fargate 启动类型的任务和网络配置指定其他 ECS 参数。随后，将使用 Fargate 启动类型运行任务的一个实例。

名为 `hello-world.yml` 的示例 Docker 复合文件：

```
version: '3'
services:
  nginx:
    image: nginx:latest
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: tutorial
        awslogs-region: us-east-1
        awslogs-stream-prefix: nginx
```

名为 `ecs-params.yml` 的示例 ECS 参数文件：

```
version: 1
task_definition:
  ecs_network_mode: awsvpc
  task_execution_role: ecsTaskExecutionRole
  task_size:
    cpu_limit: 512
    mem_limit: 2GB
  services:
    nginx:
      essential: true
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - subnet-abcd1234
        - subnet-dcba4321
      security_groups:
        - sg-abcd1234
        - sg-dcba4321
      assign_public_ip: ENABLED
```

命令:

```
ecs-cli compose --project-name hello-world --file hello-world.yml --ecs-params ecs-
params.yml up --launch-type FARGATE
```

输出：

```
INFO[0000] Using ECS task definition                     TaskDefinition=ecscompose-hello-
world:5
```

ecs-cli compose service

在 ECS 集群上利用 docker-compose 式命令管理 Amazon ECS 服务。有关 Docker 复合文件语法如何与 ECS CLI 结合使用的更多信息，请参阅使用 Docker 复合文件语法 (p. 518)。

> **Note**
>
> 要使用 Amazon ECS CLI 运行任务而不是创建服务，请参阅 ecs-cli compose (p. 472)。

ecs-cli compose service 命令将项目名称与其创建的任务定义和服务结合使用。当 Amazon ECS CLI 从复合文件创建任务定义和服务时，任务定义和服务称为 *project-name*。默认情况下，项目名称是包含 Docker 复合文件的目录的名称。但是，您也可以使用 `--project-name` 选项指定自己的项目名称。

> **Note**
>
> Amazon ECS CLI 只能管理使用 CLI 创建的任务、服务和容器实例。要管理并非使用 Amazon ECS CLI 创建的任务、服务和容器实例，请使用 AWS Command Line Interface 或 AWS 管理控制台。
>
> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli compose [--verbose] [--file *compose_file*] [--project-name *project_name*] [--task-role-arn *task_role_arn*] [--ecs-params *ecs_params_file*] [--registry-creds *value*] [--region *region*] [--cluster-config *cluster_config_name*] [--ecs-profile *ecs_profile*] [--aws-profile *aws_profile*] [--cluster *cluster_name*] [--help] service [*subcommand*] [*arguments*] [--help]

选项

名称	描述
`--verbose, --debug`	提高命令输出的详细程度以帮助诊断。 必需：否
`--file, -f` *compose-file*	指定要使用的 Docker 复合文件。此时，最新版本的 Amazon ECS CLI 支持 Docker 复合文件语法版本 1 和 2。 如果在 ecs-cli compose 运行时设置 `COMPOSE_FILE` 环境变量，则 Docker 复合文件将设置为该环境变量的值。 类型：字符串 默认值：`./docker-compose.yml` 必需：否
`--project-name, -p` *project-name*	指定要使用的项目名称。如果在运行 ecs-cli compose 时设置 `COMPOSE_PROJECT_NAME` 环境变量，则项目名称将设置为该环境变量的值。 类型：字符串 默认值：当前目录名。 必需：否
`--task-role-arn` *role_value*	指定该任务中的容器可以代入的 IAM 角色的短名称或完整的 Amazon 资源名称 (ARN)。该任务中的所有容器都将被授予在该角色中指定的权限。 类型：字符串

名称	描述
	必需：否
`--ecs-params`	指定非 Docker 复合文件固有的 ECS 参数。有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。 默认值：`./ecs-params.yml` 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

可用子命令

ecs-cli compose service 命令支持以下子命令。其中，每条子命令均具有关联的标记，这些标记可通过 `--help` 标记显示。

create

从复合文件创建 Amazon ECS 服务。将使用预期计数 0 创建服务，因此该命令不会启动任何容器。有关更多信息，请参阅ecs-cli compose service create (p. 485)。

start

在创建的 Amazon ECS 服务上启动每个容器的一个副本。此命令将服务的预期数量更新为 `1`。有关更多信息，请参阅ecs-cli compose service start (p. 490)。

up

从复合文件创建 Amazon ECS 服务（如果该服务不存在）并在集群上运行该任务的一个实例（create 和 start 的组合）。此命令将服务的预期数量更新为 `1`。有关更多信息，请参阅ecs-cli compose service up (p. 491)。

ps、list

列出集群中属于使用复合项目创建的服务的所有容器。有关更多信息，请参阅 ecs-cli compose service ps, list (p. 497)。

scale

将服务的预期数量扩展至指定计数。有关更多信息，请参阅 ecs-cli compose service scale (p. 498)。

stop

停止属于使用复合项目创建的服务的正在运行的任务。此命令将服务的预期数量更新为 `0`。有关更多信息，请参阅 ecs-cli compose service stop (p. 500)。

rm、delete、down

将服务的预期数量更新为 `0`，然后删除服务。有关更多信息，请参阅ecs-cli compose service rm, delete, down (p. 501)。

ecs-cli compose service create

从复合文件创建 Amazon ECS 服务。将使用预期计数 `0` 创建服务，因此该命令不会启动任何容器。

语法

ecs-cli compose service create [--deployment-max-percent ***n***] [--deployment-min-healthy-percent ***n***] [--load-balancer-name ***value***|--target-group-arn ***value***] [--container-name ***value***] [--container-port ***value***] [--role ***value***] [--launch-type ***launch_type***] [--health-check-grace-period ***integer***] [--create-log-groups] [--enable-service-discovery] [--vpc ***value***] [--private-dns-namespace ***value***] [--private-dns-namespace-id ***value***] [--public-dns-namespace ***value***] [--public-dns-namespace-id ***value***] [--sd-container-name ***value***] [--sd-container-port ***value***] [--dns-ttl ***value***] [--dns-type ***value***] [--healthcheck-custom-config-failure-threshold ***value***] [--scheduling-strategy ***value***] [--tags ***key1=value1,key2=value2***] [--disable-ecs-managed-tags] [--help]

选项

名称	描述
`--deployment-max-percent`	指定部署期间可在服务中保持运行状态的运行任务数的上限（服务的 `desiredCount` 的百分比）。有关更多信息，请参阅 `maximumPercent` (p. 275)。 默认值：`200` 必需：否
`--deployment-min-healthy-percent`	指定部署期间必须在服务中保持运行状态以及正常运行的运行任务数的下限（服务的 desiredCount 的百分比）。有关更多信息，请参阅 `minimumHealthyPercent` (p. 275)。 默认值：`100`

名称	描述
	必需：否
`--target-group-arn`	指定要与服务关联的之前配置的 Elastic Load Balancing 目标组的完整 Amazon 资源名称 (ARN)。 必需：否
`--container-name`	指定容器名称（与在容器定义中显示的相同）。如果指定了负载均衡器或目标组，则此参数是必需的。 必需：否，除非指定了负载均衡器或目标组。
`--container-port`	指定要与负载均衡器关联的容器上的端口。此端口必须对应于服务的任务定义中的 `containerPort`。如果指定了负载均衡器或目标组，则此参数是必需的。 必需：否，除非指定了负载均衡器或目标组。
`--load-balancer-name`	指定要与服务关联的之前配置的 Elastic Load Balancing 负载均衡器的名称。 必需：否
`--role`	指定允许 Amazon ECS 代表您调用负载均衡器或目标组的 IAM 角色的名称或完整 Amazon 资源名称 (ARN)。当您要将负载均衡器或目标组用于服务时才需要此参数。如果指定了角色参数，您还必须指定负载均衡器名称或目标组 ARN，以及容器名称和容器端口。 必需：否，除非指定了负载均衡器或目标组。
`--health-check-grace-period`	指定在任务首次启动后，Amazon ECS 服务计划程序应忽略不正常的 Elastic Load Balancing 目标运行状况检查的时长（以秒为单位）。 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--enable-service-discovery`	指定是否要为此服务启用服务发现。 必需：否
`--vpc`	指定将附加到 服务发现 的私有 DNS 命名空间的 VPC。如果指定了 `--private-dns-namespace`，则需要此参数。 必需：否
`--private-dns-namespace`	指定要用于 服务发现 的私有 DNS 命名空间的名称。如果此命名空间不存在，Amazon ECS CLI 将自动进行创建。例如，如果命名空间是 `corp`，则可以通过 DNS，用 `foo.corp` 访问名为 `foo` 的服务。如果使用此参数，您还必须使用 `--vpc` 参数指定 VPC。 必需：否

名称	描述
`--private-dns-namespace-id`	指定要用于 服务发现 的现有的私有 DNS 命名空间的 ID。如果使用此参数，您将无法指定 `--private-dns-namespace` 或 `--vpc`。 必需：否
`--public-dns-namespace`	指定要用于 服务发现 的公有 DNS 命名空间的名称。例如，如果命名空间是 `corp`，则可以通过 DNS，用 `foo.corp` 访问名为 `foo` 的服务。 必需：否
`--public-dns-namespace-id`	指定要用于 服务发现 的现有的公有 DNS 命名空间的 ID。如果使用此参数，您将无法指定 `--public-dns-namespace`。 必需：否
`--sd-container-name`	指定容器的名称，在 Docker Compose 文件中称为服务。有关更多信息，请参阅服务配置参考。如果您使用 SRV 记录，则需要此参数。 必需：否，除非使用 SRV DNS 记录。
`--sd-container-port`	指定将用于 服务发现 的容器上的端口。如果您使用 SRV 记录，则需要此参数。 必需：否，除非使用 SRV DNS 记录。
`--dns-ttl`	指定您希望 DNS 解析程序缓存用于 服务发现 的 DNS 记录设置的时长（以秒为单位）。 默认值：`60` 必需：否
`--dns-type`	指定用于 服务发现 的 DNS 记录的类型。接受的值为 `A` 或 `SRV`。如果您的任务使用 `bridge` 或 `host` 网络模式，则需要 `SRV` 记录。如果您的任务使用 `awsvpc` 网络模式，`A` 记录为默认值。 必需：否
`--healthcheck-custom-config-failure-threshold`	指定在收到 `UpdateInstanceCustomHealthStatus` 请求后，您希望 服务发现 服务在更改运行状况之前等待的 30 秒间隔数目。 默认值：`1` 必需：否

名称	描述
`--scheduling-strategy` *`value`*	指定要用于服务的计划策略。 有两种服务计划程序策略可用： • `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。 • `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。 Note Fargate 任务不支持 `DAEMON` 计划策略。 有关更多信息，请参阅 服务计划程序概念 (p. 270)。 类型：字符串 有效值：`REPLICA` | `DAEMON` 默认值：`REPLICA` 必需：否
`--tags` *`key1=value1,key2=value2`*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签使用以下格式：`key1=value1,key2=value2,key3=value3`。如果您已选择使用新的 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式，则默认情况下启用 Amazon ECS 托管标签，除非您使用 `--disable-ecs-managed-tags` 标记明确禁用它们。有关更多信息，请参阅标记资源 (p. 496)。 类型：键值对 必需：否
`--disable-ecs-managed-tags`	禁用 Amazon ECS 托管标签。有关更多信息，请参阅 标记资源以便于计费 (p. 332)。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否

名称	描述
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用负载均衡器

您可选择借助负载均衡器运行您的服务。负载均衡器将在与服务关联的各个任务间分配流量。有关更多信息，请参阅 服务负载均衡 (p. 284)。在创建服务后，在服务定义中指定的负载均衡器名称或目标组 ARN、容器名称和容器端口是不可变的。

> **Note**
>
> 您必须先创建负载均衡器资源，然后才能将服务配置为使用它们。您的负载均衡器资源应与您的容器实例位于同一 VPC 中，并且应将它们配置为使用相同的子网。您还必须向容器实例安全组添加一个安全组规则，该规则允许来自负载均衡器的入站流量。有关更多信息，请参阅 创建负载均衡器 (p. 289)。

- 要将服务配置为使用现有 Elastic Load Balancing 传统负载均衡器，您必须指定负载均衡器名称、容器名称 (显示在容器定义中) 和要从负载均衡器访问的容器端口。将此服务中的任务放在容器实例上后，容器实例将注册到在此处指定的负载均衡器。
- 要将服务配置为使用现有 Elastic Load Balancing 应用程序负载均衡器，您必须指定负载均衡器目标组 ARN、容器名称 (显示在容器定义中) 和要从负载均衡器访问的容器端口。将此服务中的任务放在容器实例上后，容器实例和端口组合将注册为在此处指定的目标组中的目标。

`--health-check-grace-period` 选项指定一个时间段，在任务首次启动后，Amazon ECS 服务计划程序在此时间段内 (以秒为单位) 将忽略不正常的 Elastic Load Balancing 目标运行状况检查。您的服务只有配置为使用负载均衡器时此设置才适用。如果您的任务需要一段时间才能启动并响应 ELB 运行状况检查，您最长可以指定 1800 秒的运行状况检查宽限期，在此期间 ECS 服务计划程序将忽略 Elastic Load Balancing 运

行状况检查状态。此宽限期可防止 Amazon ECS 服务计划程序将由于时间不足而未启动的任务标记为不正常并停止它们。

使用 服务发现

Amazon ECS 服务可以选择配置为使用 Amazon ECS 服务发现。服务发现 使用 Amazon Route 53 自动命名 API 操作来管理服务任务的 DNS 条目，使其可在 VPC 中供搜索。有关更多信息，请参阅教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 (p. 442)。

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

使用 `ecs-cli compose service create` 命令时，使用 `--tags` 标记让您可以将元数据标签添加到任务定义和服务。在创建资源时，这些标签将会添加到服务和任务定义中。这些标签将会从您的任务定义传播到服务创建的任务。如果您已选择使用新的 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式，则默认情况下启用 Amazon ECS 托管标签，除非您使用 `--disable-ecs-managed-tags` 标记明确禁用它们。有关更多信息，请参阅标记资源以便于计费 (p. 332)。

ecs-cli compose service start

在创建的 Amazon ECS 服务上启动每个容器的一个副本。此命令将服务的预期数量更新为 `1`。

语法

ecs-cli compose service start [--create-log-groups] [--force-deployment] [--help]

选项

名称	描述
`--timeout` *`value`*	指定等待运行任务计数更改的超时值（以分钟为单位，支持小数）。如果运行任务计数在指定时长内未更改，Amazon ECS CLI 将超时并返回错误。将超时时间设置为 `0` 将导致命令返回而不检查是否成功。默认超时值为 `5`（分钟）。 默认值：`5` 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--force-new-deployment`	强制实施服务的新部署。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。

名称	描述
	类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

ecs-cli compose service up

从复合文件创建 Amazon ECS 服务（如果该服务不存在）并在集群上运行该任务的一个实例（create 和 start 命令的组合）。此命令将服务的预期数量更新为 `1`。

语法

ecs-cli compose service up [--deployment-max-percent ***n***] [--deployment-min-healthy-percent ***n***] [--load-balancer-name ***value***|--target-group-arn ***value***] [--container-name ***value***] [--container-port ***value***] [--role ***value***] [--health-check-grace-period ***integer***] [--timeout ***value***] [--launch-type ***launch_type***] [--create-log-groups] [--force-deployment] [--enable-service-discovery] [--vpc ***value***] [--private-dns-namespace ***value***] [--private-dns-namespace-id ***value***] [--public-dns-namespace ***value***] [--public-dns-namespace-id ***value***] [--sd-container-name ***value***] [--sd-container-port ***value***] [--dns-ttl ***value***] [--dns-type ***value***] [--healthcheck-custom-config-failure-threshold ***value***] [--update-service-discovery] [--scheduling-strategy ***value***] [--tags ***key1=value1,key2=value2***] [--disable-ecs-managed-tags] [--help]

选项

名称	描述
`--deployment-max-percent`	指定部署期间可在服务中保持运行状态的运行任务数的上限（服务的 `desiredCount` 的百分比）。有关更多信息，请参阅 `maximumPercent` (p. 275)。 默认值：`200`

名称	描述
	必需：否
`--deployment-min-healthy-percent`	指定部署期间必须在服务中保持运行状态以及正常运行的运行任务数的下限（服务的 desiredCount 的百分比）。有关更多信息，请参阅 `minimumHealthyPercent` (p. 275)。 默认值：`100` 必需：否
`--target-group-arn`	指定要与服务关联的之前配置的 Elastic Load Balancing 目标组的完整 Amazon 资源名称 (ARN)。 必需：否
`--container-name`	指定容器名称（与在容器定义中显示的相同）。如果指定了负载均衡器或目标组，则此参数是必需的。 必需：否，除非指定了负载均衡器或目标组。
`--container-port`	指定要与负载均衡器关联的容器上的端口。此端口必须对应于服务的任务定义中的 `containerPort`。如果指定了负载均衡器或目标组，则此参数是必需的。 必需：否，除非指定了负载均衡器或目标组。
`--load-balancer-name`	指定要与服务关联的之前配置的 Elastic Load Balancing 负载均衡器的名称。 必需：否
`--role`	指定允许 Amazon ECS 代表您调用负载均衡器或目标组的 IAM 角色的名称或完整 Amazon 资源名称 (ARN)。当您要将负载均衡器或目标组用于服务时才需要此参数。如果指定了角色参数，您还必须指定负载均衡器名称或目标组 ARN，以及容器名称和容器端口。 必需：否，除非指定了负载均衡器或目标组。
`--health-check-grace-period`	指定在任务首次启动后，Amazon ECS 服务计划程序应忽略不正常的 Elastic Load Balancing 目标运行状况检查的时长（以秒为单位）。 必需：否
`--create-log-groups`	创建您的 Compose 文件中指定的 CloudWatch 日志组。 必需：否
`--enable-service-discovery`	指定是否要为此服务启用服务发现。 必需：否

名称	描述
`--vpc`	指定将附加到 服务发现 的私有 DNS 命名空间的 VPC。如果指定了 `--private-dns-namespace`，则需要此参数。 必需：否
`--private-dns-namespace`	指定要用于 服务发现 的私有 DNS 命名空间的名称。如果此命名空间不存在，Amazon ECS CLI 将自动进行创建。例如，如果命名空间是 `corp`，则可以通过 DNS，用 `foo.corp` 访问名为 `foo` 的服务。如果使用此参数，您还必须使用 `--vpc` 参数指定 VPC。 必需：否
`--private-dns-namespace-id`	指定要用于 服务发现 的现有的私有 DNS 命名空间的 ID。如果使用此参数，您将无法指定 `--private-dns-namespace` 或 `--vpc`。 必需：否
`--public-dns-namespace`	指定要用于 服务发现 的公有 DNS 命名空间的名称。例如，如果命名空间是 `corp`，则可以通过 DNS，用 `foo.corp` 访问名为 `foo` 的服务。 必需：否
`--public-dns-namespace-id`	指定要用于 服务发现 的现有的公有 DNS 命名空间的 ID。如果使用此参数，您将无法指定 `--public-dns-namespace`。 必需：否
`--sd-container-name`	指定容器的名称，在 Docker Compose 文件中称为服务。有关更多信息，请参阅服务配置参考。如果您使用 SRV 记录，则需要此参数。 必需：否，除非使用 SRV DNS 记录。
`--sd-container-port`	指定将用于 服务发现 的容器上的端口。如果您使用 SRV 记录，则需要此参数。 必需：否，除非使用 SRV DNS 记录。
`--dns-ttl`	指定您希望 DNS 解析程序缓存用于 服务发现 的 DNS 记录设置的时长（以秒为单位）。 默认值：`60` 必需：否
`--dns-type`	指定用于 服务发现 的 DNS 记录的类型。接受的值为 `A` 或 `SRV`。如果您的任务使用 `bridge` 或 `host` 网络模式，则需要 `SRV` 记录。如果您的任务使用 `awsvpc` 网络模式，`A` 记录为默认值。 必需：否

名称	描述
`--healthcheck-custom-config-failure-threshold`	指定在收到 `UpdateInstanceCustomHealthStatus` 请求后，您希望 服务发现 服务在更改运行状况之前等待的 30 秒间隔数目。 默认值：`1` 必需：否
`--update-service-discovery`	如果指定，将导致更新针对 `--dns-ttl` 和 `--healthcheck-custom-config-failure-threshold` 的 服务发现 服务设置。 必需：否
`--scheduling-strategy` *`value`*	指定要用于服务的计划策略。 有两种服务计划程序策略可用： • `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。 • `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。 Note Fargate 任务不支持 `DAEMON` 计划策略。 有关更多信息，请参阅 服务计划程序概念 (p. 270)。 类型：字符串 有效值：`REPLICA` | `DAEMON` 默认值：`REPLICA` 必需：否

名称	描述
`--tags` *`key1=value1,key2=value2`*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签使用以下格式：`key1=value1,key2=value2,key3=value3`。如果您已选择使用新的 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式，则默认情况下启用 Amazon ECS 托管标签，除非您使用 `--disable-ecs-managed-tags` 标记明确禁用它们。有关更多信息，请参阅标记资源 (p. 496)。 类型：键值对 必需：否
`--disable-ecs-managed-tags`	禁用 Amazon ECS 托管标签。有关更多信息，请参阅 标记资源以便于计费 (p. 332)。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用负载均衡器

您可选择借助负载均衡器运行您的服务。负载均衡器将在与服务关联的各个任务间分配流量。有关更多信息，请参阅 服务负载均衡 (p. 284)。在创建服务后，在服务定义中指定的负载均衡器名称或目标组 ARN、容器名称和容器端口是不可变的。

> **Note**
>
> 您必须先创建负载均衡器资源，然后才能将服务配置为使用它们。您的负载均衡器资源应与您的容器实例位于同一 VPC 中，并且应将它们配置为使用相同的子网。您还必须向容器实例安全组添加一个安全组规则，该规则允许来自负载均衡器的入站流量。有关更多信息，请参阅 创建负载均衡器 (p. 289)。

- 要将服务配置为使用现有 Elastic Load Balancing 传统负载均衡器，您必须指定负载均衡器名称、容器名称 (显示在容器定义中) 和要从负载均衡器访问的容器端口。将此服务中的任务放在容器实例上后，容器实例将注册到在此处指定的负载均衡器。
- 要将服务配置为使用现有 Elastic Load Balancing 应用程序负载均衡器，您必须指定负载均衡器目标组 ARN、容器名称 (显示在容器定义中) 和要从负载均衡器访问的容器端口。将此服务中的任务放在容器实例上后，容器实例和端口组合将注册为在此处指定的目标组中的目标。

`--health-check-grace-period` 选项指定一个时间段，在任务首次启动后，Amazon ECS 服务计划程序在此时间段内 (以秒为单位) 将忽略不正常的 Elastic Load Balancing 目标运行状况检查。您的服务只有配置为使用负载均衡器时此设置才适用。如果您的任务需要一段时间才能启动并响应 ELB 运行状况检查，您最长可以指定 1800 秒的运行状况检查宽限期，在此期间 ECS 服务计划程序将忽略 Elastic Load Balancing 运行状况检查状态。此宽限期可防止 Amazon ECS 服务计划程序将由于时间不足而未启动的任务标记为不正常并停止它们。

使用 服务发现

Amazon ECS 服务可以选择配置为使用 Amazon ECS 服务发现。服务发现 使用 Amazon Route 53 自动命名 API 操作来管理服务任务的 DNS 条目，使其可在 VPC 中供搜索。有关更多信息，请参阅教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 (p. 442)。

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

使用 `ecs-cli compose service up` 命令时，使用 `--tags` 标记让您可以将元数据标签添加到任务定义和服务。在创建资源时，这些标签将会添加到服务和任务定义中。这些标签将会从您的任务定义传播到服务创建的任务。如果您已选择使用新的 Amazon 资源名称 (ARN) 和资源标识符 (ID) 格式，则默认情况下启用 Amazon ECS 托管标签，除非您使用 `--disable-ecs-managed-tags` 标记明确禁用它们。有关更多信息，请参阅标记资源以便于计费 (p. 332)。

示例

示例 1

此示例将启动 `hello-world.yml` 复合文件中的项目名称为 `hello-world` 的 Amazon ECS 服务。

```
ecs-cli compose --project-name hello-world --file hello-world.yml service up
```

输出:

```
INFO[0000] Using ECS task definition                     TaskDefinition="ecscompose-hello-
world:7"
INFO[0000] Created an ECS service                        service=ecscompose-service-hello-
world taskDefinition="ecscompose-hello-world:7"
```

```
INFO[0000] Updated ECS service successfully              desiredCount=1
 serviceName=ecscompose-service-hello-world
INFO[0015] (service ecscompose-service-hello-world) has started 1 tasks: (task
 682dc22f-8bfa-4c28-b6f8-3a916bd8f86a).  timestamp=2017-08-18 21:16:00 +0000 UTC
INFO[0060] Service status                                desiredCount=1 runningCount=1
 serviceName=ecscompose-service-hello-world
INFO[0060] ECS Service has reached a stable state        desiredCount=1 runningCount=1
 serviceName=ecscompose-service-hello-world
```

示例 2

此示例将从 `nginx-compose.yml` 复合文件创建服务并将其配置为使用现有 应用程序负载均衡器。

```
ecs-cli compose -f nginx-compose.yml service up --target-group-arn
 arn:aws:elasticloadbalancing:us-east-1:aws_account_id:targetgroup/ecs-cli-
alb/9856106fcc5d4be8 --container-name nginx --container-port 80 --role ecsServiceRole
```

示例 3

此示例将从 `nginx-compose.yml` 复合文件创建服务并将其配置为使用运行状况检查宽限期为 25 秒的现有应用程序负载均衡器。

```
ecs-cli compose -f nginx-compose.yml service up --target-group-arn
 arn:aws:elasticloadbalancing:us-east-1:aws_account_id:targetgroup/ecs-cli-
alb/9856106fcc5d4be8 --container-name nginx --container-port 80 --role ecsServiceRole --
health-check-grace-period 25
```

ecs-cli compose service ps, list

列出集群中属于使用复合项目创建的服务的所有容器。

语法

ecs-cli compose service ps|list [--desired-status *status*] [--help]

选项

名称	描述
`--desired-status` *`status`*	筛选容器列表结果时使用的容器所需状态。 必需：否 有效值：`RUNNING` | `STOPPED`
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否

名称	描述
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

ecs-cli compose service scale

将服务的预期数量扩展至指定计数。

语法

ecs-cli compose service scale [--deployment-max-percent ***n***] [--deployment-min-healthy-percent ***n***] [--timeout ***value***] ***n*** [--help]

选项

名称	描述
`--deployment-max-percent`	指定部署期间可在服务中保持运行状态的运行任务数的上限（服务的 `desiredCount` 的百分比）。有关更多信息，请参阅 `maximumPercent` (p. 275)。 默认值：`200` 必需：否
`--deployment-min-healthy-percent`	指定部署期间必须在服务中保持运行状态以及正常运行的运行任务数的下限（服务的 desiredCount 的百分比）。有关更多信息，请参阅 `minimumHealthyPercent` (p. 275)。 默认值：`100` 必需：否

名称	描述
`--timeout` *`value`*	指定等待运行任务计数更改的超时值（以分钟为单位，支持小数）。如果运行任务计数在指定时长内未更改，Amazon ECS CLI 将超时并返回错误。将超时时间设置为 0 将导致命令返回而不检查是否成功。默认超时值为 5（分钟）。 默认值：5 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例 1

此示例将由 `hello-world` 项目创建的服务扩展至预期数量 2。

```
ecs-cli compose --project-name hello-world --file hello-world.yml service scale 2
```

输出:

```
INFO[0000] Updated ECS service successfully              desiredCount=2
 serviceName=ecscompose-service-hello-world
INFO[0000] Service status                                desiredCount=2 runningCount=1
 serviceName=ecscompose-service-hello-world
INFO[0030] (service ecscompose-service-hello-world) has started 1 tasks: (task
 80602da8-442c-48ea-a8a9-80328c302b89).  timestamp=2017-08-18 21:17:44 +0000 UTC
INFO[0075] Service status                                desiredCount=2 runningCount=2
 serviceName=ecscompose-service-hello-world
INFO[0075] ECS Service has reached a stable state        desiredCount=2 runningCount=2
 serviceName=ecscompose-service-hello-world
```

ecs-cli compose service stop

停止属于使用复合项目创建的服务的正在运行的任务。此命令将服务的预期数量更新为 0。

`--timeout` 选项指定等待运行任务计数更改的超时值（以分钟为单位，支持小数）。如果运行任务计数在指定时长内未更改，Amazon ECS CLI 将超时并返回错误。将超时时间设置为 0 将导致命令返回而不检查是否成功。默认超时值为 5（分钟）。

语法

ecs-cli compose service stop [--timeout ***value***] [--help]

选项

名称	描述
`--timeout` *value*	指定等待运行任务计数更改的超时值（以分钟为单位，支持小数）。如果运行任务计数在指定时长内未更改，Amazon ECS CLI 将超时并返回错误。将超时时间设置为 0 将导致命令返回而不检查是否成功。默认超时值为 5（分钟）。 默认值：5 必需：否
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *cluster_config_name*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否

名称	描述
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

ecs-cli compose service rm, delete, down

将服务的预期数量更新为 `0`，然后删除服务。

语法

此命令在使用时接受 `rm`、`delete` 或 `down`。

ecs-cli compose service rm|delete|down [--timeout ***value***] [--delete-namespace] [--help]

选项

名称	描述
`--timeout` *`value`*	指定等待运行任务计数更改的超时值（以分钟为单位，支持小数）。如果运行任务计数在指定时长内未更改，Amazon ECS CLI 将超时并返回错误。将超时时间设置为 `0` 将导致命令返回而不检查是否成功。默认超时值为 `5`（分钟）。 默认值：`5` 必需：否
`--delete-namespace`	如果指定，将删除使用 compose service create 或 compose service up 命令创建的私有命名空间。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。

名称	描述
	类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例 1

此示例将由 `hello-world` 项目创建的服务扩展至预期数量，然后删除服务。0

```
ecs-cli compose --project-name hello-world --file hello-world.yml service rm
```

输出:

```
INFO[0000] Updated ECS service successfully              desiredCount=0
 serviceName=ecscompose-service-hello-world
INFO[0000] Service status                                desiredCount=0 runningCount=2
 serviceName=ecscompose-service-hello-world
INFO[0015] Service status                                desiredCount=0 runningCount=0
 serviceName=ecscompose-service-hello-world
INFO[0015] (service ecscompose-service-hello-world) has stopped 2 running tasks: (task
 682dc22f-8bfa-4c28-b6f8-3a916bd8f86a) (task 80602da8-442c-48ea-a8a9-80328c302b89).
 timestamp=2017-08-18 21:25:28 +0000 UTC
INFO[0015] ECS Service has reached a stable state        desiredCount=0 runningCount=0
 serviceName=ecscompose-service-hello-world
INFO[0015] Deleted ECS service                           service=ecscompose-service-hello-
world
INFO[0015] ECS Service has reached a stable state        desiredCount=0 runningCount=0
 serviceName=ecscompose-service-hello-world
```

ecs-cli logs

从 CloudWatch Logs 中检索容器日志。仅对使用 `awslogs` 驱动程序且指定了日志流前缀的任务有效。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

```
ecs-cli logs --task-id task_id [--task-def task_definition] [--follow]
[--filter-pattern search_string] [--since n_minutes] [--start-time
2006-01-02T15:04:05+07:00] [--end-time 2006-01-02T15:04:05+07:00] [--
timestamps] [--help]
```

选项

名称	描述
`--task-id` *`task_id`*	打印此 ECS 任务的日志。 类型：字符串 必需：是
`--task-def` *`task_definition`*	指定与任务 ID 关联的 ECS 任务定义的名称或完整 Amazon 资源名称 (ARN)。仅当任务停止后需要。 类型：字符串 必需：否
`--follow`	指定是否应流式传输日志。 必需：否
`--filter-pattern` *`search_string`*	指定要在日志中搜索的子字符串。 类型：字符串 必需：否
`--since` *`n`*	返回比相对持续时间短 (以分钟计) 的日志。不能与 `--start-time` 一起使用。 类型：整数 必需：否
`--start-time` *`timestamp`*	返回特定日期之后的日志 (格式：RFC 3339。示例：`2006-01-02T15:04:05+07:00`)。不能与 `--since` 标记一起使用。 必需：否
`--end-time` *`timestamp`*	返回特定日期之前的日志 (格式：RFC 3339。示例：`2006-01-02T15:04:05+07:00`)。不能与 `--follow` 一起使用。

名称	描述
	必需：否
`--timestamps`	指定是否在日志输出中的每一行上显示时间戳。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例将打印任务的日志。

```
ecs-cli logs --task-id task_id
```

如果成功，日志的内容将位于输出中。

ecs-cli check-attributes

检查容器实例的给定列表是否可以通过选中某给定任务定义的属性来运行该定义。该任务定义所需但不存在于容器实例上的输出属性。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

```
ecs-cli check-attributes [--task-def task_definition] [--container-instances
value] [--help]
```

选项

名称	描述
`--task-def` *`task_definition`*	指定与任务 ID 关联的 ECS 任务定义的名称或完整 Amazon 资源名称 (ARN)。仅当任务停止后需要。 类型：字符串 必需：否
`--container-instances` *`value`*	容器实例 ID 或完整 ARN 条目的列表，用于检查所有必需的属性是否适用于 RunTask 的任务定义。 必需：否
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否

名称	描述
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

示例

此示例检查多个容器实例并验证它们是否包含要成功运行指定任务定义所需的属性。

```
ecs-cli check-attributes --container-
instances 28c5abd2-360e-41a0-81d8-0afca2d08d9b,45510138-f24f-47c6-a418-71c46dd51f88 --
cluster default --region us-east-2 --task-def fluentd-test
```

输出：

```
Container Instance                    Missing Attributes
28c5abd2-360e-41a0-81d8-0afca2d08d9b  com.amazonaws.ecs.capability.logging-driver.fluentd
45510138-f24f-47c6-a418-71c46dd51f88  None
```

ecs-cli registry-creds

有助于在 Amazon ECS 中创建和使用私有注册表凭证。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

`ecs-cli registry-creds [--region` *`region`*`] [--cluster-config` *`cluster_config_name`*`] [--ecs-profile` *`ecs_profile`*`] [--aws-profile` *`aws_profile`*`] [--help] [`*`subcommand`*`] [`*`arguments`*`] [--help]`

选项

名称	描述
`--region, -r` *`region`*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的集群。 类型：字符串

名称	描述
	必需：否
`--cluster-config` *`cluster_config_name`*	指定要使用的 Amazon ECS 集群配置的名称。默认为作为默认值设置的集群设置。 类型：字符串 必需：否
`--ecs-profile` *`ecs_profile`*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *`aws_profile`*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--cluster, -c` *`cluster_name`*	指定要使用的 Amazon ECS 集群名称。默认为使用 configure 命令配置的集群。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

可用子命令

ecs-cli registry-creds 命令支持以下子命令。其中，每条子命令均具有关联的标记，这些标记可通过 `--help` 标记显示。

up

生成 AWS Secrets Manager 密钥和 IAM 任务执行角色以在 Amazon ECS 任务定义中使用。有关更多信息，请参阅 ecs-cli registry-creds up (p. 507)。

help

显示指定命令的帮助文本。

ecs-cli registry-creds up

生成 AWS Secrets Manager 密钥和 IAM 任务执行角色以在 Amazon ECS 任务定义中使用。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli registry-creds up ***./creds_input_file.yml*** --role-name ***value*** [--update-existing-secrets] [--no-role] [--no-output-value] [--output-dir ***value***] [--tags ***key1=value1,key2=value2***] [--help]

选项

名称	描述
./creds_input_file.yml	指定与私有注册表身份验证相关的值。有关更多信息，请参阅 使用私有注册表身份验证 (p. 508)。 必需：是
`--role-name` *value*	用于新任务执行角色的名称。如果该角色已存在，新策略将附加到现有角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色 (p. 406)。 Note 我们建议创建特定于每个应用程序的新任务执行角色以避免向不需要密钥的应用程序授予对密钥的权限。 必需：是
`--update-existing-secrets`	指定是否应使用新凭证值更新现有密钥。 必需：否
`--no-role`	如果已指定，则不会创建任何任务执行角色。 必需：否
`--no-output-file`	如果已指定，则不会创建用于 `compose` 的输出文件。 必需：否
`--output-dir` *value*	应在其中创建输出文件的目录。如果没有指定，则在当前工作目录中创建该文件。 必需：否
`--tags` *key1=value1,key2=value2*	指定要用于您的 AWS 资源的元数据。每一个标签都包含一个键和一个可选值。标签采用以下格式：`key1=value1,key2=value2,key3=value3`。有关更多信息，请参阅 标记资源 (p. 509)。 类型：键值对 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

使用私有注册表身份验证

当使用 `ecs-cli registry-creds up` 命令来管理您的私有注册表身份验证凭证时，使用输入文件来指定某些字段。在使用此命令时，您必须指定输入文件的文件名或路径。

目前，该文件支持以下架构：

```
version: 1
registry_credentials:
  registry_name:
    secrets_manager_arn: string
    username: string
    password: string
    kms_key_id: string
    container_names:
      - string
```

以下是对其中每个字段的描述。

registry_name

在创建新密钥或更新现有密钥时用作密钥名称。密钥名称应仅包含 ASCII 字母、数字或以下任意字符：/_+=.@-。Amazon ECS CLI 向密钥名称添加前缀以指示它是由 CLI 创建的。有关更多信息，请参阅 CreateSecret。

必需：否

secrets_manager_arn

现有密钥的完整 ARN。用于指定或更新现有密钥。必须采用以下格式：

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret_name
```

必需：否

username

指定私有注册表的用户名。我们建议对用户名使用环境变量，以确保输入文件中没有存储任何敏感信息。当使用环境变量时，请使用格式 `${VAR_NAME}`。

必需：否

password

指定私有注册表的密码。我们建议对密码使用环境变量，以确保输入文件中没有存储任何敏感信息。当使用环境变量时，请使用格式 `${VAR_NAME}`。

必需：否

kms_key_id

指定要用于加密密钥的 AWS KMS 客户主密钥 (CMK) 的 ARN、密钥 ID 或别名。有关更多信息，请参阅 CreateSecret。

必需：否

container_names

对应于 Docker 复合文件中的服务名称。有关更多信息，请参阅 ecs-cli compose (p. 472) 或 ecs-cli compose service (p. 482)。

必需：否

标记资源

Amazon ECS CLI 支持以资源标签的形式将元数据添加到您的 AWS 资源。每一个标签都包含一个键和一个可选值。资源标签可用于成本分配、自动化和访问控制。有关更多信息，请参阅 AWS 标记策略。

使用 `ecs-cli registry-creds up` 命令时，使用 `--tags` 标记让您可以将元数据标签添加到 Secrets Manager 密钥，然后添加到 IAM 角色。

> **Note**
>
> 将标记您账户内的现有 Secrets Manager 密钥，但只有 IAM 角色才能在创建过程中进行标记。如果您使用的是现有 IAM 角色，则无法添加新标签。

示例

使用私有注册表身份验证凭证创建密钥

此示例使用 `creds_input.yml` 输入文件中指定的私有注册表凭证创建密钥。

创建名为 `creds_input.yml` 的私有注册表凭证文件，其中包含私有注册表的用户名和密码以及将使用私有注册表凭证的容器的名称。我们建议对凭证使用环境变量，以确保输入文件中没有存储任何敏感信息。此文件中的容器名称对应于 Docker 复合文件中的服务名称 `database`。

```
version: '1'
registry_credentials:
  dockerhub:
    username: ${MY_REPO_USERNAME}
    password: ${MY_REPO_PASSWORD}
    container_names:
      - database
```

> **Important**
>
> 我们建议对密码使用环境变量，以确保输入文件中没有存储任何敏感信息。如果您的输入文件包含敏感信息，请确保您在使用后将其删除。

创建密钥。此命令使用输入文件中的名称（在此示例中为 `dockerhub`）创建密钥。Amazon ECS CLI 向密钥名称添加前缀以指示它是由 CLI 创建的。还指定您的任务执行角色的名称。

```
ecs-cli registry-creds up ./creds_input.yml --role-name secretsTaskExecutionRole
```

输出：

```
INFO[0000] Processing credentials for registry dockerhub...
INFO[0000] New credential secret created:
 arn:aws:secretsmanager:region:aws_account_id:secret:amazon-ecs-cli-setup-dockerhub-VeDqXm
INFO[0000] Creating resources for task execution role ecsTaskExecutionRole...
INFO[0000] Created new task execution role arn:aws:iam::aws_account_id:role/
ecsTaskExecutionRole
INFO[0000] Created new task execution role policy arn:aws:iam::aws_account_id:policy/
amazon-ecs-cli-setup-bugBashRole-policy-20181023T210805Z
INFO[0000] Attached AWS managed policy arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy to role ecsTaskExecutionRole
INFO[0001] Attached new policy arn:aws:iam::aws_account_id:policy/amazon-ecs-cli-setup-
bugBashRole-policy-20181023T210805Z to role ecsTaskExecutionRole
INFO[0001] Writing registry credential output to new file C:\Users\brandejo\regcreds
\regCredTest\ecs-registry-creds_20181023T210805Z.yml
```

通过此命令创建一个输出文件，其中包含任务执行角色名称、创建的密钥的 ARN 和容器名称。在使用 ecs-cli compose 或 ecs-cli compose service 命令时，使用 `--registry-creds` 选项指定此文件。有关更多信息，请参阅 ecs-cli compose (p. 472) 或 ecs-cli compose service (p. 482)。

下面是一个示例输出文件：

```
version: "1"
```

```
registry_credential_outputs:
  task_execution_role: secretsTaskExecutionRole
  container_credentials:
    dockerhub:
      credentials_parameter: arn:aws:secretsmanager:region:aws_account_id:secret:amazon-
ecs-cli-setup-dockerhub-bbHiEk
      container_names:
      - database
```

通过使用 KMS 密钥的私有注册表身份验证凭证创建密钥

此示例通过 `creds_input.yml` 输入文件中指定的使用 KMS 密钥加密的私有注册表凭证创建密钥。

创建名为 `creds_input.yml` 的私有注册表凭证文件，其中包含私有注册表的用户名和密码以及将使用私有注册表凭证的容器的名称。我们建议对凭证使用环境变量，以确保输入文件中没有存储任何敏感信息。在存储密钥时，指定的 KMS 密钥 ARN 会对值进行加密。此文件中的容器名称对应于 Docker 复合文件中的服务名称 `database`。

```
version: '1'
registry_credentials:
  dockerhub:
    username: ${MY_REPO_USERNAME}
    password: ${MY_REPO_PASSWORD}
    kms_key_id: kmsKeyARN
    container_names:
      - database
```

> **Important**
>
> 我们建议对密码使用环境变量，以确保输入文件中没有存储任何敏感信息。如果您的输入文件包含敏感信息，请确保您在使用后将其删除。

为多个私有注册表创建多个密钥

此示例使用私有注册表凭证为多个注册表创建多个密钥。

创建名为 `creds_input.yml` 的私有注册表凭证文件，其中包含来自两个不同私有注册表的凭证。每组凭证用于创建自己的密钥。此示例还显示了使用一个密钥的两个不同的容器。

```
version: '1'
registry_credentials:
  dockerhub:
    username: ${MY_REPO_USERNAME}
    password: ${MY_REPO_PASSWORD}
    container_names:
      - prod
      - dev
  quay.io:
    username: ${MY_REPO_USERNAME}
    password: ${MY_REPO_PASSWORD}
    container_names:
      - database
```

> **Important**
>
> 我们建议对密码使用环境变量，以确保输入文件中没有存储任何敏感信息。如果您的输入文件包含敏感信息，请确保您在使用后将其删除。

ecs-cli local

通过从 Amazon ECS 任务定义创建 Docker Compose 文件来本地运行 Amazon ECS 任务。

Important

所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli local [*subcommand*] [*arguments*] [--help]

选项

名称	描述
`--region, -r` *region*	指定要使用的 AWS 区域。默认为使用 configure 命令配置的区域。 类型：字符串 必需：否
`--ecs-profile` *ecs_profile*	指定要使用的 Amazon ECS 配置文件的名称。默认为使用 configure profile 命令配置的配置文件。 类型：字符串 必需：否
`--aws-profile` *aws_profile*	指定要使用的 AWS 配置文件。使您能够使用 `~/.aws/credentials` 中现有指定配置文件中的 AWS 凭证。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

可用子命令

ecs-cli compose 命令支持以下子命令。其中，每条子命令均具有关联的标记，这些标记可通过 `--help` 标记显示。

create

从 Amazon ECS 任务定义创建 Docker Compose 文件。有关更多信息，请参阅 ecs-cli local create (p. 513)。

up

从 Amazon ECS 任务定义本地运行容器。有关更多信息，请参阅 ecs-cli local up (p. 514)。

down

停止和删除本地运行的容器。有关更多信息，请参阅 ecs-cli local down (p. 516)。

ps

列出本地运行的容器。有关更多信息，请参阅 ecs-cli local ps (p. 517)。

help

显示指定命令的帮助文本。

ecs-cli local create

从 Amazon ECS 任务定义创建 Docker Compose 文件。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli local create [--task-def-file *filename*] [--task-def-remote *value*] [--output *output_file*]

选项

名称	描述
`--task-def-file` *filename*	指定包含要转换为 Docker Compose 文件的任务定义 JSON 的文件名。如果未指定一个这样的文件名，ECS CLI 将在当前目录中查找名为 `task-definition.json` 的文件。 类型：JSON 必需：否
`--task-def-remote` *value*	指定要转换为 Docker Compose 文件的任务定义的完整 Amazon 资源名称 (ARN) 或 family:revision。如果指定没有修订的任务定义系列，则使用最新修订。 类型：字符串 必需：否
`--output` *output_file*	指定要将 Docker Compose 文件写入的本地文件名。如果未指定一个这样的文件名，则默认为 `docker-compose.local.yml`。 类型：字符串 必需：否
`--help, -h`	显示指定命令的帮助文本。 必需：否

示例

从本地 JSON 文件创建 Docker Compose 文件

此示例从包含 Amazon ECS 任务定义的本地 JSON 文件创建 Docker Compose 文件。

```
ecs-cli local create --task-def-file task-definition.json
```

输出:

```
INFO[0000] Successfully wrote docker-compose.ecs-local.yml
INFO[0000] Successfully wrote docker-compose.ecs-local.override.yml
```

从远程任务定义创建 Docker Compose 文件

此示例从名为 `hello-world` 的最新修订版的 Amazon ECS 任务定义创建 Docker Compose 文件。

```
ecs-cli local create --task-def-remote hello-world
```

输出:

```
INFO[0000] Successfully wrote docker-compose.ecs-local.yml
INFO[0000] Successfully wrote docker-compose.ecs-local.override.yml
```

ecs-cli local up

从 Amazon ECS 任务定义本地运行容器。默认情况下，此命令会在当前目录中查找名为 `task-definition.json` 的任务定义 JSON 文件。如果该任务定义文件不存在，则必须使用某个 `--task-def` 选项指定一个任务定义文件，如下所述。此命令还会在运行容器之前创建 `--output` 选项中指定的本地 Docker Compose 文件。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli local create up [--task-def-compose *filename*] [--task-def-file *filename*] [--task-def-remote *value*] [--output *output_file*] [--override *filename*]

选项

名称	描述
`--task-def-compose` *filename*	指定要本地运行的 Docker Compose 文件。 类型：字符串 必需：否
`--task-def-file` *filename*	指定要本地运行的任务定义 JSON 文件。如果未指定一个这样的文件，ECS CLI 将在当前目录中查找名为 `task-definition.json` 的文件。 类型：字符串 必需：否
`--task-def-remote` *value*	指定要转换为 Docker Compose 文件的任务定义的完整 Amazon 资源名称 (ARN) 或 family:revision。如果指定没有修订的任务定义系列，则使用最新修订。 类型：字符串 必需：否

名称	描述
`--output` *`output_file`*	指定要将 Docker Compose 文件写入的本地文件名。如果未指定一个这样的文件名，则默认为 `docker-compose.local.yml`。如果输出文件已存在，CLI 将提示一条覆盖请求。 类型：字符串 必需：否
`--override` *`filename`*	指定要使用的本地 Docker Compose 覆盖文件名。 类型：字符串 必需：否

示例

从本地任务定义 JSON 文件本地运行容器

此示例本地运行在名为 `hello-world.json` 的本地任务定义文件中定义的容器。

```
ecs-cli local create --task-def-file hello-world.json
```

输出:

```
INFO[0001] Successfully wrote docker-compose.ecs-local.yml
INFO[0002] Successfully wrote docker-compose.ecs-local.override.yml
INFO[0002] The network ecs-local-network already exists
INFO[0002] The amazon-ecs-local-container-endpoints container already exists with ID
 5976522f4cafb840e5f003a2285fc439ed1b2a89aa74634958c6a6105ca6edd1
INFO[0002] Started container with ID
 5976522f4cafb840e5f003a2285fc439ed1b2a89aa74634958c6a6105ca6edd1
INFO[0002] Using docker-compose.ecs-local.yml, docker-compose.ecs-local.override.yml files
 to start containers
Compose out: Found orphan containers (downloads_httpd_1) for this project. If you removed
 or renamed this service in your compose file, you can run this command with the --remove-
orphans flag to clean it up.
Creating downloads_simple-app_1 ... done
```

从远程任务定义本地运行容器

此示例本地运行在名为 `hello-world` 的最新修订版的 Amazon ECS 任务定义中定义的容器。

```
ecs-cli local up --task-def-remote hello-world
```

输出:

```
INFO[0000] Reading task definition from hello-world

INFO[0002] Successfully wrote docker-compose.ecs-local.yml
INFO[0004] Successfully wrote docker-compose.ecs-local.override.yml
INFO[0004] The network ecs-local-network already exists
INFO[0005] The amazon-ecs-local-container-endpoints container already exists with ID
 5976522f4cafb840e5f003a2285fc439ed1b2a89aa74634958c6a6105ca6edd1
```

```
INFO[0005] Started container with ID
 5976522f4cafb840e5f003a2285fc439ed1b2a89aa74634958c6a6105ca6edd1
INFO[0005] Using docker-compose.ecs-local.yml, docker-compose.ecs-local.override.yml files
 to start containers
Compose out: Found orphan containers (downloads_httpd_1) for this project. If you removed
 or renamed this service in your compose file, you can run this command with the --remove-
orphans flag to clean it up.
Creating downloads_hello-world_1 ... done
```

ecs-cli local down

停止和删除本地运行的容器。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli local down [--task-def-file *filename*] [--task-def-remote *task_definition_ARN_family*] [--all]

选项

名称	描述
`--task-def-file` *filename*	停止和删除所有与任务定义文件名匹配的运行中的容器。如果忽略 `--task-def-file` 和 `--task-def-remote`，则 ECS CLI 默认为 `task-definition.json`。 类型：字符串 必需：否
`--task-def-remote` *value*	停止和删除所有与指定的任务定义 Amazon 资源名称 (ARN) 或 family:revision 匹配的运行中的容器。如果指定没有修订的任务定义系列，则使用最新修订。 类型：字符串 必需：否
`--all`	停止和删除所有本地运行的容器。 类型：字符串 必需：否

示例

停止本地运行的容器

此示例停止使用 `hello-world.json` 任务定义文件的本地运行的容器。

```
ecs-cli local down --task-def-file hello-world.json
```

输出:

```
INFO[0000] Stop and remove 1 container(s)
INFO[0011] Stopped container with id 9df4c584d905
INFO[0011] Removed container with id 9df4c584d905
INFO[0011] The network ecs-local-network has no more running tasks
INFO[0012] Stopped container with name amazon-ecs-local-container-endpoints
INFO[0012] Removed container with name amazon-ecs-local-container-endpoints
INFO[0012] Removed network with name ecs-local-network
```

停止所有本地运行的容器

此示例停止所有本地运行的容器。

```
ecs-cli local down --all
```

ecs-cli local ps

列出本地运行的容器。

> **Important**
>
> 所介绍的某些功能仅可用于最新版本的 ECS CLI。要获取最新版本，请参阅安装 Amazon ECS CLI (p. 425)。

语法

ecs-cli local ps [--task-def-file *filename*] [--task-def-remote *value*] [--all] [--json]

选项

名称	描述
`--task-def-file` *filename*	列出所有与任务定义文件名匹配的本地运行的容器。如果忽略 `--task-def-file` 和 `--task-def-remote`，则 ECS CLI 默认为 `task-definition.json`。 类型：字符串 必需：否
`--task-def-remote` *value*	列出所有与任务定义 Amazon 资源名称 (ARN) 或 family:revision 匹配的运行中的容器。如果指定没有修订的任务定义系列，则使用最新修订。 类型：字符串 必需：否
`--all`	列出所有本地运行的容器。 类型：字符串 必需：否
`--json`	将输出设置为 JSON 格式。 类型：字符串

名称	描述
	必需：否

示例

列出所有本地运行的容器

本示例列出所有本地运行的容器。

```
ecs-cli local ps --all
```

输出:

```
CONTAINER ID        IMAGE               STATUS              PORTS               NAMES
                TASKDEFINITION
9df4c584d905        httpd:2.4           Up 15 seconds       0.0.0.0:80->80/tcp   /
downloads_simple-app_1   /Users/brandejo/Downloads/task-definition.json
```

列出所有使用指定的任务定义文件的本地运行的容器

本示例列出使用 `hello-world.json` 任务定义的本地运行的容器。

```
ecs-cli local ps --task-def-file hello-world.json
```

使用 Docker 复合文件语法

`ecs-cli compose` 和 `ecs-cli compose service` 命令允许您使用 Docker Compose 文件创建任务定义并管理 Amazon ECS 任务和服务。有关更多信息，请参阅 ecs-cli compose (p. 472) 和 ecs-cli compose service (p. 482)。

目前，最新版本的 Amazon ECS CLI 仅支持主要版本的 Docker Compose 文件语法（版本 1、2 和 3）。Compose 文件中指定的版本必须为字符串 `"1"`、`"1.0"`、`"2"`、`"2.0"`、`"3"` 或 `"3.0"`。Docker Compose 次要版本不受支持。

默认情况下，Amazon ECS CLI 命令将在当前目录中查找一个名为 `docker-compose.yml` 的 Docker Compose 文件。但是，您也可以使用 `--file` 选项指定 Compose 文件的其他文件名或路径。这在使用 Amazon ECS CLI 一次性管理多个 Compose 文件中的任务和服务时特别有用。

Amazon ECS CLI 的复合文件中支持以下参数：

- `cap_add` (对使用 Fargate 启动类型的任务无效)
- `cap_drop` (对使用 Fargate 启动类型的任务无效)
- `command`
- `cpu_shares`

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `cpu_shares`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。

- `devices` (对使用 Fargate 启动类型的任务无效)
- `dns`

- `dns_search`
- `entrypoint`
- `environment`：如果复合文件中未指定环境变量值，但该值存在于 shell 环境中，则 shell 环境变量值将传递到为任何关联的任务或服务创建的任务定义。

  > **Important**
  >
  > 建议不要对敏感信息 (如凭证数据) 使用纯文本环境变量。

- `env_file`

  > **Important**
  >
  > 建议不要对敏感信息 (如凭证数据) 使用纯文本环境变量。

- `extra_hosts`
- `healthcheck`（仅限 Compose 文件版本 3）

  > **Note**
  >
  > 使用复合文件时不支持 `start_period` 字段。要指定 `start_period`，请使用 `ecs-params.yml`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。

- `hostname`
- `image`
- `labels`
- `links` (对使用 Fargate 启动类型的任务无效)
- `log_driver`（仅限 Compose 文件版本 1）
- `log_opt`（仅限 Compose 文件版本 1）
- `logging`（Compose 文件版本 2 和 3）
  - `driver`
  - `options`
- `mem_limit` (以字节为单位)

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `mem_limit`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。

- `mem_reservation` (以字节为单位)

  > **Note**
  >
  > 如果您使用 Compose 版本 3 格式，应在 `ecs-params.yml` 文件中指定 `mem_reservation`。 有关更多信息，请参阅 使用 Amazon ECS 参数 (p. 520)。

- `ports`
- `privileged` (对使用 Fargate 启动类型的任务无效)
- `read_only`
- `security_opt`
- `shm_size`（仅限 Compose 文件版本 1 和 2，并且对使用 Fargate 启动类型的任务无效）
- `tmpfs` (对使用 Fargate 启动类型的任务无效)
- `ulimits`
- `user`
- `volumes`
- `volumes_from`（仅限 Compose 文件版本 1 和 2）

- `working_dir`

  Important

  此时不支持 `build` 指令。

有关 Docker 复合文件语法的更多信息，请参阅 Docker 文档中的复合文件参考。

使用 Amazon ECS 参数

在使用 `ecs-cli compose` 或 `ecs-cli compose service` 命令管理 Amazon ECS 任务和服务时，Amazon ECS 任务定义中存在与 Docker 复合文件中的字段不对应的特定字段。您可以通过 `--ecs-params` 标志使用 ECS 参数文件指定这些值。默认情况下，此命令将在当前目录中查找一个名为 `ecs-params.yml` 的 ECS 参数文件。但是，您也可以使用 `--ecs-params` 选项指定 ECS 参数文件的其他文件名或路径。

目前，该文件支持以下架构：

```
version: 1
task_definition:
  ecs_network_mode: string
  task_role_arn: string
  task_execution_role: string
  task_size:
    cpu_limit: string
    mem_limit: string
  pid_mode: string
  ipc_mode: string
  services:
    <service_name>:
      essential: boolean
      repository_credentials:
        credentials_parameter: string
      cpu_shares: integer
      mem_limit: string
      mem_reservation: string
      gpu: string
      init_process_enabled: boolean
      healthcheck:
        test: ["CMD", "curl -f http://localhost"]
        interval: string
        timeout: string
        retries: integer
        start_period: string
      secrets:
        - value_from: string
          name: string
  docker_volumes:
      - name: string
        scope: string
        autoprovision:
        driver: string
        driver_opts: boolean
           string: string
        labels:
           string: string
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - subnet_id1
```

```
        - subnet_id2
      security_groups:
        - secgroup_id1
        - secgroup_id2
      assign_public_ip: ENABLED
  task_placement:
    strategy:
      - type: string
        field: string
    constraints:
      - type: string
         expression: string
  service_discovery:
    container_name: string
    container_port: integer
    private_dns_namespace:
        vpc: string
        id: string
        name: string
        description: string
    public_dns_namespace:
        id: string
        name: string
    service_discovery_service:
        name: string
        description: string
        dns_config:
          type: string
          ttl: integer
        healthcheck_custom_config:
          failure_threshold: integer
```

`task_definition` 下列出的字段对应于要包含在您的 Amazon ECS 任务定义中的字段。

- `ecs_network_mode` – 对应于 ECS 任务定义中的 `networkMode`。支持的值为 `none`、`bridge`、`host` 或 `awsvpc`。默认值为 `bridge`。如果您使用的是任务联网，则此字段必须设置为 `awsvpc`。有关更多信息，请参阅网络模式 (p. 52)。
- `task_role_arn` — 要与任务关联的 IAM 角色的名称或完整 ARN。有关更多信息，请参阅任务角色 (p. 52)。
- `task_execution_role` – 任务执行角色的名称或完整 ARN。如果您希望任务能够将容器应用程序日志存储在 CloudWatch 中或允许您的任务从 Amazon ECR 拉取容器映像，则此字段是必填字段。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。
- `task_size` – 任务的 CPU 和内存值。如果使用的是 EC2 启动类型，则此字段是可选的，并且可使用任何值。如果使用的是 Fargate 启动类型，此字段是必填的，并且您必须对 `cpu` 和 `memory` 参数使用以下几组值。

CPU value	Memory value (MiB)
256 (.25 vCPU)	512 (0.5GB), 1024 (1GB), 2048 (2GB)
512 (.5 vCPU)	1024 (1GB), 2048 (2GB), 3072 (3GB), 4096 (4GB)
1024 (1 vCPU)	2048 (2GB), 3072 (3GB), 4096 (4GB), 5120 (5GB), 6144 (6GB), 7168 (7GB), 8192 (8GB)
2048 (2 vCPU)	Between 4096 (4GB) and 16384 (16GB) in increments of 1024 (1GB)
4096 (4 vCPU)	Between 8192 (8GB) and 30720 (30GB) in increments of 1024 (1GB)

有关更多信息，请参阅任务大小 (p. 77)。

- `pid_mode` – 用于任务中的容器的过程命名空间。有效值为 `host` 或 `task`。如果指定了 `host`，则在同一容器实例上指定了 `host` PID 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的 IPC 资源。如果指定了 `task`，则指定任务中的所有容器将共享相同的过程命名空间。如果未指定任何值，则默认值为私有命名空间。有关更多信息，请参阅 Docker 运行参考 中的 PID 设置。

  如果使用了 `host` PID 模式，请注意，发生非预期的过程命名空间公开的风险会提高。有关更多信息，请参阅 Docker 安全性。

  > **Note**
  >
  > 使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

- `ipc_mode` – 用于任务中的容器的 IPC 资源命名空间。有效值为 `host`、`task` 或 `none`。如果指定了 `host`，则在同一容器实例上指定了 `host` IPC 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的 IPC 资源。如果指定了 `task`，则指定任务中的所有容器将共享相同的 IPC 资源。如果指定了 `none`，则任务的容器中的 IPC 资源是私有的，不与任务中或容器实例上的其他容器共享。如果未指定任何值，则 IPC 资源命名空间共享取决于容器实例上的 Docker 守护程序设置。有关更多信息，请参阅 Docker 运行参考 中的 IPC 设置。

  如果使用了 `host` IPC 模式，请注意，发生非预期的 IPC 命名空间公开的风险会提高。有关更多信息，请参阅 Docker 安全性。

  如果您使用 `systemControls` 为任务中的容器设置具有命名空间的内核参数，则以下内容将适用于您的 IPC 资源命名空间。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的系统控制。

  - 对于使用 `host` IPC 模式的任务，不支持与 IPC 命名空间相关的 `systemControls`。
  - 对于使用 `task` IPC 模式的任务，与 IPC 命名空间相关的 `systemControls` 将适用于任务中的所有容器。

  > **Note**
  >
  > 使用 Fargate 启动类型的 Windows 容器或任务不支持此参数。

- `services` – 对应于您的 Docker 复合文件中列出的服务，其中 `service_name` 与要运行的容器的名称匹配。其字段将合并到容器定义中。
  - `essential` – 如果将容器的 `essential` 参数标记为 `true`，并且该容器出于任何原因发生故障或停止，则属于此任务的所有其他容器将停止。如果将容器的 `essential` 参数标记为 `false`，则容器发生故障不会影响任务中的剩余容器。默认值为 `true`。

    所有任务都必须具有至少一个主要容器。如果您有一个由多个容器组成的应用程序，则应将用于相同目的的容器分成多个组件，然后将不同的组件分为多个任务定义。
  - `repository_credentials` – 如果您使用私有存储库拉取映像，`repository_credentials` 让您为包含私有存储库凭证的私有密钥的名称指定一个 AWS Secrets Manager 密钥 ARN 作为 `credential_parameter`。有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。
  - `cpu_shares` –此参数映射到 Docker 复合文件参考中的 `cpu_shares`。如果您使用的是 Docker compose 版本 3，则此字段是可选的，并且必须在 ECS params 文件而不是复合文件中指定。在 Docker compose 版本 2 中，此字段可以在复合文件或 ECS params 文件中指定。如果在 ECS params 文件中指定它，则值将覆盖复合文件中存在的值。
  - `mem_limit` –此参数映射到 Docker 复合文件参考中的 `mem_limit`。如果您使用的是 Docker compose 版本 3，则此字段是可选的，并且必须在 ECS params 文件而不是复合文件中指定。在 Docker compose 版本 2 中，此字段可以在复合文件或 ECS params 文件中指定。如果在 ECS params 文件中指定它，则值将覆盖复合文件中存在的值。
  - `mem_reservation` –此参数映射到 Docker 复合文件参考中的 `mem_reservation`。如果您使用的是 Docker compose 版本 3，则此字段是可选的，并且必须在 ECS params 文件而不是复合文件中指定。在 Docker compose 版本 2 中，此字段可以在复合文件或 ECS params 文件中指定。如果在 ECS params 文件中指定它，则值将覆盖复合文件中存在的值。

- gpu – Amazon ECS 容器代理将为容器预留的物理 GPU 的数量。此参数将映射到任务定义中的 resourceRequirements 字段。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。
- init_process_enabled – 此参数让您能够在容器内运行可转发信号并获得进程的 init 进程。此参数将映射到 docker run 的 --init 选项。

  此参数要求您容器实例上的 Docker Remote API 版本为 1.25 或更高。
- healthcheck –此参数映射到 Docker 复合文件参考中的 healthcheck。test 字段也可指定为 command，并且必须是字符串或列表。如果它是列表，则第一个项目必须是 NONE、CMD 或 CMD-SHELL。如果它是字符串，则等同于指定 CMD-SHELL 后跟该字符串。interval、timeout 和 start_period 字段指定为字符串格式的持续时间。例如：2.5s、10s、1m30s、2h23m 或 5h34m56s。

  Note

  如果未指定任何单位，则假定为秒。例如，可以指定 10s 或只指定 10。
- secrets – 此参数使您可以向容器中注入敏感数据，方法是将您的敏感数据存储在 AWS Systems Manager Parameter Store 参数中，然后在容器定义中引用它们。有关更多信息，请参阅 指定敏感数据 (p. 107)。
  - value_from – 这是要向容器公开的 AWS Systems Manager Parameter Store ARN 或名称。如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用密钥的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。
  - name – 要在容器上设置为环境变量的值。
- docker_volumes – 此参数允许您创建 Docker 卷。name 键是必需的，scope、autoprovision、driver、driver_opts 和 labels 与任务定义中的 Docker 卷配置字段相对应。有关更多信息，请参阅 Amazon Elastic Container Service API Reference 中的 DockerVolumeConfiguration。使用 docker_volumes 键定义的卷可以按名称在复合文件中引用（即使未在复合文件中指定）。

run_params 下列出的字段针对所需的值，因为任何 API 调用的选项未与任务定义明确相关联，例如 compose up (RunTask) 和 compose service up (CreateService)。目前，run_params 下唯一受支持的参数为 network_configuration，该参数在使用任务联网时和使用 Fargate 启动类型的任务时是必需参数。

- network_configuration – 如果您为 ecs_network_mode 指定 awsvpc，则是必需的。它使用一个带以下子字段的嵌套参数 awsvpc_configuration：
  - subnets – 要与您的任务关联的已用子网 ID 的列表。列出的子网必须与要在其上启动任务的实例位于同一 VPC 和可用区中。
  - security_groups – 要与您的任务关联的安全组 ID 的列表。列出的安全性必须与要在其上启动任务的实例位于同一 VPC 中。
  - assign_public_ip – 此字段支持的值为 ENABLED 或 DISABLED。此字段仅用于使用 Fargate 启动类型的任务。如果此字段存在于将任务联网与 EC2 启动类型结合使用的任务中，则请求将失败。
- task_placement – 此参数允许您指定任务放置选项。如果使用的是 EC2 启动类型，则是可选的。Fargate 启动类型不支持。有关更多信息，请参阅Amazon ECS 任务放置 (p. 255)。

  它具有以下子字段：
  - strategy – 对象列表，带有 2 个键。有效键为 type 和 field。
    - type – 有效值为 random、binpack 或 spread。如果指定了 random，则不应提供 field 键。
    - field – 有效值取决于策略类型。
      - 对于 spread，有效值为 instanceId、host 或属性键-值对，例如 attribute:ecs.instance-type =~ t2.*。
      - 对于 binpack，有效值为 cpu 或 memory。
  - constraints – 对象列表，带有 2 个键。有效键为 type 和 expression。

  - `type` – 有效值为 `distinctInstance` 和 `memberOf`。如果指定了 `distinctInstance`，则不应提供 `expression` 键。
  - `expression` – 当类型为 `memberOf` 时，有效值为属性或任务组的键-值对，例如 `task:group == databases` 或 `attribute:color =~ green`。
- `service_discovery` – 此参数可用于使用 Amazon Route 53 自动命名 API 操作配置 Amazon ECS 服务发现以管理服务的任务的 DNS 条目。有关更多信息，请参阅教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 (p. 442)。

将 AWS CLI 与 Amazon ECS 结合使用

AWS 命令行界面 (CLI) 是用于管理 AWS 服务的统一工具。只通过一个工具进行下载和配置，您就可以使用命令行控制多个 AWS 服务并利用脚本来自动执行这些服务。有关 AWS CLI 的更多信息，请参阅 http://aws.amazon.com/cli/。

有关可用于管理 AWS 资源的其他工具的更多信息，包括不同的 AWS 开发工具包、IDE 工具包和 Windows PowerShell 命令行工具，请参阅 http://aws.amazon.com/tools/。

以下步骤帮助您使用 Fargate 或 EC2 任务设置 Amazon ECS 集群：

主题

- 教程：使用 AWS CLI 创建包含 Fargate 任务的集群 (p. 525)
- 教程：使用 AWS CLI 创建包含 EC2 任务的集群 (p. 531)

教程：使用 AWS CLI 创建包含 Fargate 任务的集群

以下步骤帮助您在 Amazon ECS 中使用 AWS CLI 设置集群、注册任务定义、运行任务和执行其他常见方案。请确保您使用的是最新版本的 AWS CLI。有关如何升级到最新版本的更多信息，请参阅安装 AWS 命令行界面。

主题

- 先决条件 (p. 525)
- 第 1 步：（可选）创建集群 (p. 526)
- 第 2 步：注册任务定义 (p. 526)
- 第 3 部：列出任务定义 (p. 528)
- 步骤 4：创建服务 (p. 528)
- 步骤 5：列出服务 (p. 529)
- 步骤 6：描述正在运行的服务 (p. 530)

先决条件

本教程假设以下先决条件已完成：

- 安装并配置了最新版本的 AWS CLI。有关安装或升级 AWS CLI 的更多信息，请参阅安装 AWS 命令行界面。
- Amazon ECS 的设置 (p. 7) 中的步骤已完成。
- 您的 AWS 用户具有 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。有关更多信息，请参阅教程：为您的集群创建带有公有和私有子网的 VPC。

第 1 步：（可选）创建集群

默认情况下，您的账户会收到一个 `default` 集群。

> **Note**
>
> 使用为您提供的 `default` 集群的好处是您不必在后续命令中指定 `--cluster` *`cluster_name`* 选项。如果您自行创建非默认集群，您必须为您打算用于该集群的每个命令指定 `--cluster` *`cluster_name`*。

使用以下命令自行创建具有唯一名称的集群：

```
aws ecs create-cluster --cluster-name fargate-cluster
```

输出:

```
{
    "cluster": {
        "status": "ACTIVE",
        "statistics": [],
        "clusterName": "fargate-cluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

第 2 步：注册任务定义

您必须先注册任务定义才能在 ECS 集群上运行任务。任务定义是分组在一起的一系列容器。下面的示例是创建 PHP Web 应用程序的一个简单任务定义。有关可用任务定义参数的更多信息，请参阅 Amazon ECS 任务定义 (p. 43)。

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "httpd:2.4",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!
</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></
html>' >  /usr/local/apache2/htdocs/index.html && httpd-foreground\""
```

```
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

上述示例 JSON 可通过两种方式传递到 AWS CLI：您可以将任务定义 JSON 保存为文件并使用 `--cli-input-json file://path_to_file.json` 选项传递它。您也可以对 JSON 中的引号进行转义并在命令行上传递 JSON 容器定义，如以下示例中所示。如果您选择在命令行上传递容器定义，您的命令还需要一个 `--family` 参数，该参数用于使任务定义的多个版本保持互相关联。

将 JSON 文件用于容器定义：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

register-task-definition 将在其完成注册后返回任务定义的说明。

```
{
    "taskDefinition": {
        "status": "ACTIVE",
        "networkMode": "awsvpc",
        "family": "sample-fargate",
        "placementConstraints": [],
        "requiresAttributes": [
            {
                "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
            },
            {
                "name": "ecs.capability.task-eni"
            }
        ],
        "cpu": "256",
        "compatibilities": [
            "EC2",
            "FARGATE"
        ],
        "volumes": [],
        "memory": "512",
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "taskDefinitionArn": "arn:aws:ecs:region:aws_account_id:task-definition/sample-
fargate:2",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "fargate-app",
                "mountPoints": [],
                "image": "httpd:2.4",
                "cpu": 0,
                "portMappings": [
                    {
                        "protocol": "tcp",
                        "containerPort": 80,
                        "hostPort": 80
                    }
                ],
                "entryPoint": [
                    "sh",
```

```
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!
</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></
html>' >  /usr/local/apache2/htdocs/index.html && httpd-foreground\""
                ],
                "essential": true,
                "volumesFrom": []
            }
        ],
        "revision": 2
    }
}
```

第 3 部：列出任务定义

您可以随时使用 list-task-definitions 命令列出您的账户的任务定义。此命令的输出将显示 `family` 和 `revision` 值，您可以在调用 run-task 或 start-task 时将这些值一起使用。

```
aws ecs list-task-definitions
```

输出:

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:1",
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:2"
    ]
}
```

步骤 4：创建服务

为账户注册任务后，您可以为集群中已注册的任务创建服务。在此示例中，您将创建一个服务，其中至少有 2 个 `sample-fargate:1` 任务定义实例在您的集群中保持运行。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 2 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}"
```

输出:

```
{
    "service": {
        "status": "ACTIVE",
        "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/sample-
fargate:1",
        "pendingCount": 0,
        "launchType": "FARGATE",
        "loadBalancers": [],
        "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/ecs.amazonaws.com/
AWSServiceRoleForECS",
        "placementConstraints": [],
        "createdAt": 1510811361.128,
        "desiredCount": 2,
        "networkConfiguration": {
```

```
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-abcd1234"
                ],
                "securityGroups": [
                    "sg-abcd1234"
                ],
                "assignPublicIp": "DISABLED"
            }
        },
        "platformVersion": "LATEST",
        "serviceName": "fargate-service",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
        "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "status": "PRIMARY",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-abcd1234"
                        ],
                        "securityGroups": [
                            "sg-abcd1234"
                        ],
                        "assignPublicIp": "DISABLED"
                    }
                },
                "pendingCount": 0,
                "launchType": "FARGATE",
                "createdAt": 1510811361.128,
                "desiredCount": 2,
                "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1",
                "updatedAt": 1510811361.128,
                "platformVersion": "0.0.1",
                "id": "ecs-svc/9223370526043414679",
                "runningCount": 0
            }
        ],
        "events": [],
        "runningCount": 0,
        "placementStrategy": []
    }
}
```

步骤 5：列出服务

列出您的集群的服务。您应看到您在上一部分中创建的服务。您可以选取从此命令返回的服务名称或完整 ARN 并在稍后将其用于描述服务。

```
aws ecs list-services --cluster fargate-cluster
```

输出:

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-service"
```

```
    ]
}
```

步骤 6：描述正在运行的服务

使用之前检索到的服务名称描述服务，以获取有关任务的更多信息。

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

输出:

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/sample-
fargate:1",
            "pendingCount": 2,
            "launchType": "FARGATE",
            "loadBalancers": [],
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "placementConstraints": [],
            "createdAt": 1510811361.128,
            "desiredCount": 2,
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "DISABLED"
                }
            },
            "platformVersion": "LATEST",
            "serviceName": "fargate-service",
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            },
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "DISABLED"
                        }
                    },
                    "pendingCount": 2,
                    "launchType": "FARGATE",
                    "createdAt": 1510811361.128,
                    "desiredCount": 2,
```

```
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1",
                    "updatedAt": 1510811361.128,
                    "platformVersion": "0.0.1",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "placementStrategy": []
        }
    ],
    "failures": []
}
```

教程：使用 AWS CLI 创建包含 EC2 任务的集群

以下步骤帮助您在 Amazon ECS 中使用 AWS CLI 设置集群、注册任务定义、运行任务和执行其他常见方案。请确保您使用的是最新版本的 AWS CLI。有关如何升级到最新版本的更多信息，请参阅安装 AWS 命令行界面。

主题

- 先决条件 (p. 531)
- 第 1 步：（可选）创建集群 (p. 532)
- 第 2 步：使用 Amazon ECS AMI 启动实例 (p. 532)
- 第 3 步：列出容器实例 (p. 533)
- 第 4 步：描述容器实例 (p. 534)
- 第 5 步：注册任务定义 (p. 536)
- 第 6 部：列出任务定义 (p. 537)
- 第 7 步：运行任务 (p. 537)
- 第 8 步：列出任务 (p. 538)
- 第 9 步：描述正在运行的任务 (p. 538)

先决条件

本教程假设以下先决条件已完成：

- 安装并配置了最新版本的 AWS CLI。有关安装或升级 AWS CLI 的更多信息，请参阅安装 AWS 命令行界面。
- Amazon ECS 的设置 (p. 7) 中的步骤已完成。
- 您的 AWS 用户具有 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。有关更多信息，请参阅教程：为您的集群创建带有公有和私有子网的 VPC。

第 1 步：（可选）创建集群

默认情况下，当您启动第一个容器实例时，您的账户将收到一个 `default` 集群。

> **Note**
>
> 使用为您提供的 `default` 集群的好处是您不必在后续命令中指定 `--cluster` *`cluster_name`* 选项。如果您自行创建非默认集群，您必须为您打算用于该集群的每个命令指定 `--cluster` *`cluster_name`*。

使用以下命令自行创建具有唯一名称的集群：

```
aws ecs create-cluster --cluster-name MyCluster
```

输出:

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

第 2 步：使用 Amazon ECS AMI 启动实例

您的集群中必须有一个 Amazon ECS 容器实例，您才能在该集群上运行任务。如果您的集群中没有任何容器实例，请参阅启动 Amazon ECS 容器实例 (p. 165)以了解更多信息。

下表按区域列出了当前的 Amazon ECS-optimized Amazon Linux AMI ID。

Region	AMI Name	AMI ID	EC2 Console Link
us-east-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0151b45908571e14c	Launch instance
us-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0c09d65d2051ada93	Launch instance
us-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0b4f28359911d5896	Launch instance
us-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-077368b501184adb9	Launch instance
ap-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0eeaf74d330e7e8ca	Launch instance

Region	AMI Name	AMI ID	EC2 Console Link
ap-northeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-084d4626fec6f3ca3	Launch instance
ap-northeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-03c5d1f1ba6ec1b7d	Launch instance
ap-south-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-09f6bc049f323302c	Launch instance
ap-southeast-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-01813416d767d88db	Launch instance
ap-southeast-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-07fcafbb9f3a4f7db	Launch instance
ca-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-011742239f8f42f46	Launch instance
eu-central-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0bceb1887b6b37130	Launch instance
eu-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0d7012b0cae33c045	Launch instance
eu-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-02b9d5ef54d57fb7d	Launch instance
eu-west-2	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0f49b2a901463508b	Launch instance
eu-west-3	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0db8d6e7dcf3cb362	Launch instance
sa-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0ad0be326813dda96	Launch instance
us-gov-east-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-033b28d8ef5fe3f73	Launch instance
us-gov-west-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-b0bafdd1	Launch instance
cn-north-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-041f913b515304af9	Launch instance
cn-northwest-1	amzn-ami-2018.03.v-amazon-ecs-optimized	ami-0be42f3b41b17b513	Launch instance

第 3 步：列出容器实例

在启动您的容器实例后的几分钟内，Amazon ECS 代理将向您的默认集群注册该实例。您可以通过运行以下命令列出集群中的容器实例：

```
aws ecs list-container-instances --cluster default
```

输出:

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

第 4 步：描述容器实例

在拥有某个容器实例的 ARN 或 ID 后，您就可以使用 describe-container-instances 命令获取有关该实例的有价值的信息，例如剩余的和已注册的 CPU 和内存资源。

```
aws ecs describe-container-instances --cluster default --container-
instances container_instance_ID
```

输出:

```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "ec2InstanceId": "instance_id",
            "agentConnected": true,
            "containerInstanceArn": "arn:aws:ecs:us-west-2:aws_account_id:container-
instance/container_instance_ID",
```

```
            "pendingTasksCount": 0,
            "remainingResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "runningTasksCount": 0,
            "attributes": [
                {
                    "name": "com.amazonaws.ecs.capability.privileged-container"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
                }
            ],
            "versionInfo": {
                "agentVersion": "1.5.0",
                "agentHash": "b197edd",
                "dockerVersion": "DockerVersion: 1.7.1"
            }
        }
    ]
```

```
}
```

您也可以在 Amazon EC2 控制台中或使用 aws ec2 describe-instances --instance-id *instance_id* 命令查找可用于监控实例的 Amazon EC2 实例 ID。

第 5 步：注册任务定义

您必须先注册任务定义才能在 ECS 集群上运行任务。任务定义是分组在一起的一系列容器。以下示例是一个简单的任务定义，它将使用 Docker Hub 中的 `busybox` 映像并休眠 360 秒。有关可用任务定义参数的更多信息，请参阅 Amazon ECS 任务定义 (p. 43)。

```
{
  "containerDefinitions": [
    {
      "name": "sleep",
      "image": "busybox",
      "cpu": 10,
      "command": [
        "sleep",
        "360"
      ],
      "memory": 10,
      "essential": true
    }
  ],
  "family": "sleep360"
}
```

上述示例 JSON 可通过两种方式传递到 AWS CLI：您可以将任务定义 JSON 保存为文件并使用 `--cli-input-json file://`*`path_to_file.json`* 选项传递它。您也可以对 JSON 中的引号进行转义并在命令行上传递 JSON 容器定义，如以下示例中所示。如果您选择在命令行上传递容器定义，您的命令还需要一个 `--family` 参数，该参数用于使任务定义的多个版本保持互相关联。

将 JSON 文件用于容器定义：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/sleep360.json
```

将 JSON 字符串用于容器定义：

```
aws ecs register-task-definition --family sleep360 --container-definitions "[{\"name\":
\"sleep\",\"image\":\"busybox\",\"cpu\":10,\"command\":[\"sleep\",\"360\"],\"memory\":10,
\"essential\":true}]"
```

register-task-definition 将在其完成注册后返回任务定义的说明。

```
{
    "taskDefinition": {
        "volumes": [],
        "taskDefinitionArn": "arn:aws:ec2:us-east-1:aws_account_id:task-definition/
sleep360:1",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "sleep",
                "mountPoints": [],
                "image": "busybox",
                "cpu": 10,
                "portMappings": [],
                "command": [
```

```
                    "sleep",
                    "360"
                ],
                "memory": 10,
                "essential": true,
                "volumesFrom": []
            }
        ],
        "family": "sleep360",
        "revision": 1
    }
}
```

第 6 部：列出任务定义

您可以随时使用 list-task-definitions 命令列出您的账户的任务定义。此命令的输出将显示 `family` 和 `revision` 值，您可以在调用 run-task 或 start-task 时将这些值一起使用。

```
aws ecs list-task-definitions
```

输出:

```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```

第 7 步：运行任务

为您的账户注册任务并启用注册到您的集群的容器实例后，您可以在您的集群中运行已注册的任务。在本实例中，您将 `sleep360:1` 任务定义的单个实例放置在默认集群中。

```
aws ecs run-task --cluster default --task-definition sleep360:1 --count 1
```

输出:

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "PENDING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
```

```
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "PENDING",
                    "name": "sleep"
                }
            ]
        }
    ]
}
```

第 8 步：列出任务

列出您的集群的任务。您应看到您在上一部分中运行的任务。您可以选取从此命令返回的 ID 或完整 ARN 并在稍后将其用于描述任务。

```
aws ecs list-tasks --cluster default
```

输出:

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```

第 9 步：描述正在运行的任务

使用之前检索到的任务 ID 描述任务，以获取有关任务的更多信息。

```
aws ecs describe-tasks --cluster default --task task_ID
```

输出:

```
{
    "failures": [],
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "RUNNING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
```

```
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "RUNNING",
                    "name": "sleep",
                    "networkBindings": []
                }
            ]
        }
    ]
}
```

Amazon ECS 服务限制

下表提供了针对 AWS 账户的 Amazon ECS 服务限制（也称作配额），可以更改这些限制。有关可用于 Amazon ECS 的其他 AWS 服务（如 Elastic Load Balancing 和 Auto Scaling）的服务限制的更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务限制。

资源	默认限制
每个区域、每个账户的集群数	2000
每个集群的容器实例数	2000
每个集群的服务数	1000
每个服务的任务数 (预期数量)	1000
每个服务的 Elastic Load Balancing 目标组数	5
每个区域每个账户对应的使用 Fargate 启动类型的任务数	50
对于每个区域、每个账户，使用 Fargate 启动类型的任务的公有 IP 地址数	50

下表提供了无法更改的其他 Amazon ECS 限制。

资源	限制
每个服务的 传统负载均衡器 数	1
每个 run-task 启动的任务数 (`count`)	10
每个 start-task 的容器实例数	10
每个任务定义系列的修订数 Note 取消注册任务定义修订不会将其排除在此限制之外。	1000000
任务定义大小限制	32 KiB
任务定义最大容器	10
子网的数量在 `awsvpcConfiguration` 中指定	16
安全组的数量在 `awsvpcConfiguration` 中指定	5
使用 Fargate 启动类型的任务中的多个容器使用的共享卷的最大大小	4GB
使用 Fargate 启动类型的任务的最大容器存储	10GB
每个资源（任务、服务、任务定义、集群和容器实例）的最大标签数	50

Amazon ECS 中的常见使用案例

本主题将提供 Amazon ECS: 微服务和批处理作业中的常见使用案例的指南。在这里，您可以找到对于让您的应用程序在 Amazon ECS 上运行可能很有用的注意事项和外部资源以及每个解决方案的常见方面。

主题

- 微服务 (p. 541)
- 批处理作业 (p. 542)

微服务

微服务是使用一种软件架构性方法构建的，该方法可将复杂的应用程序分解为小型、独立的服务。容器非常适合用于运行小型、独立的服务，并且它们还具有以下优点：

- 借助容器，服务可以使用您的所有依赖项在不可变的映像中轻松建模。
- 容器可使用任何应用程序和任何编程语言。
- 容器映像是一个受版本控制的项目，因此您可以跟踪容器映像到其来源。
- 您可以本地测试容器并大规模部署同一项目。

以下部分将介绍您在设计要在 Amazon ECS 上运行的微服务架构时必须考虑的一些方面和难题。您也可以在 GitHub 上查看微服务参考架构。有关更多信息，请参阅使用 Amazon ECS、AWS CloudFormation 和 应用程序负载均衡器 部署微服务。

主题

- Auto Scaling (p. 541)
- 服务发现 (p. 541)
- 授权和密钥管理 (p. 542)
- 日志记录 (p. 542)
- 持续集成和持续部署 (p. 542)

Auto Scaling

您的微服务架构的应用程序负载可能随时间而变化。响应式应用程序可以扩展或缩减，具体取决于实际或预期的负载。Amazon ECS 为您提供了多个工具，这些工具不仅可扩展在集群中运行的服务，还可扩展实际集群本身。

例如，Amazon ECS 为您的集群和服务提供了 CloudWatch 指标。有关更多信息，请参阅Amazon ECS CloudWatch 指标 (p. 341)。您可以监控集群和服务的内存和 CPU 利用率。然后，使用这些指标触发 CloudWatch 警报，这些警报可在您的集群资源几乎耗尽时自动扩展集群。在您不需要很多资源时缩减集群。有关更多信息，请参阅教程：通过 CloudWatch 警报扩展容器实例 (p. 349)。

除了增大集群大小之外，您还可以选择将 Amazon ECS 服务配置为使用 服务 Auto Scaling 根据 CloudWatch 警报上调或下调其预期数量。服务 Auto Scaling 在支持 Amazon ECS 的所有区域中可用。有关更多信息，请参阅服务 Auto Scaling (p. 299)。

服务发现

服务发现是大多数分布式系统和服务导向型架构的关键组件。利用服务发现，当您的微服务组件在给定基础设施上创建和终止时，系统将自动发现它们。有几种可让您的服务变得可发现的方法。以下资源介绍了几个示例：

- 使用 Amazon EC2 Container Service 和 Application Load Balancer 运行容器化微服务：本文介绍如何使用 Elastic Load Balancing 应用程序负载均衡器 的动态端口映射和基于路径的路由功能来为微服务架构提供服务发现。
- Amazon Elastic Container Service - 参考架构：服务发现：此 Amazon ECS 参考架构向使用 CloudWatch Events、Lambda 和 Route 53 私有托管区域的容器提供服务发现。
- 通过 Consul 与 Amazon ECS 实现的服务发现：本文展示名为 Consul by HashiCorp 的第三方工具如何通过为 ECS 集群提供服务发现来增强 Amazon ECS 的功能（辅以示例应用程序）。

授权和密钥管理

管理密钥 (如应用程序的数据库凭证) 始终是一个具有挑战性的问题。使用 Parameter Store 和适用于任务的 IAM 角色管理 Amazon ECS 应用程序的密钥一文重点介绍了如何将 Amazon ECS 的适用于任务的 IAM 角色 (p. 418)功能与 AWS Systems Manager Parameter Store 集成。Parameter Store 提供了一个可管理配置数据的集中式存储，无论数据是纯文本数据（如数据库字符串）还是密钥（如通过 AWS Key Management Service 加密的密码）均适用。

日志记录

您可以将容器实例配置为向 CloudWatch Logs 发送日志信息。这使您能够在一个方便位置查看容器实例中的不同日志。有关在您的利用Amazon ECS-optimized AMI 启动的容器实例上开始使用 CloudWatch Logs 的更多信息，请参阅将 CloudWatch Logs 用于容器实例 (p. 181)。

您可在任务中配置容器以将日志信息发送到 CloudWatch Logs。这使您能够在一个方便的位置查看容器中的不同日志，并防止您的容器日志占用您的容器实例上的磁盘空间。有关在您的任务定义中开始使用 `awslogs` 日志驱动程序的更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。

持续集成和持续部署

持续集成和持续部署 (CICD) 是一个用于基于 Docker 容器的微服务架构的常见流程。您可以创建执行以下操作的管道：

- 监控对源代码存储库的更改
- 从该源构建新的 Docker 映像
- 将映像推送到映像存储库，如 Amazon ECR 或 Docker Hub
- 将 Amazon ECS 服务更新为在您的应用程序中使用新映像

以下资源将概述如何以不同的方式执行此操作：

- ECS 参考架构：持续部署：此参数架构将展示如何使用 CodePipeline、CodeBuild 和 AWS CloudFormation 来实现应用程序到 Amazon ECS 的持续部署。
- 使用 Jenkins、GitHub 和 Amazon ECR 为 Amazon ECS 实现持续交付管道：此 AWS 实验室存储库帮助您使用 Jenkins、GitHub 和 Amazon ECR 为 Amazon ECS 设置和配置持续交付管道。
- 适用于容器应用程序的管道让使用 mu 变得轻松：AWS 开放源代码博客上的这篇文章说明了如何使用 mu，以通过 Amazon ECS、CodePipeline 和 CodeBuild 为容器工作负载配置持续的传输管道。

批处理作业

Docker 容器特别适合批处理作业工作负载。批处理作业通常是短暂且易并行的。您可以将批处理应用程序打包到 Docker 映像中，以便在任何位置 (如在 Amazon ECS 任务中) 部署它。如果您有兴趣运行批处理作业工作负载，请考虑以下资源：

- AWS Batch：对于任何规模的完全托管的批处理，您都应考虑使用 AWS Batch。AWS Batch 让开发人员、科学家和工程师能够轻松高效地在 AWS 上运行数十万个批处理计算任务。AWS Batch 可根据提交的批处理任务的数量和特定资源要求动态预置最佳的计算资源（如 CPU 或内存优化实例）数量和类型。有关更多信息，请参阅 AWS Batch 产品详细信息页。
- Amazon ECS 参考架构：批处理：此参考架构展示如何使用 AWS CloudFormation、Amazon S3、Amazon SQS 和 CloudWatch 警报来处理 Amazon ECS 上的批处理。

AWS App Mesh 和 Amazon ECS 入门

AWS App Mesh 是一种基于 Envoy 代理的服务网格，可轻松监控和控制微服务。App Mesh 将标准化微服务进行通信的方式，从而为您提供端到端可见性并有助于确保您的应用程序的高可用性。

App Mesh 将为您提供对应用程序中的每个微服务的一致可见性和网络流量控制。有关更多信息，请参阅 AWS App Mesh 用户指南。

本主题可帮助您将 AWS App Mesh 与在 Amazon ECS 上运行的现有微服务应用程序结合使用。

先决条件

App Mesh 支持对其组件使用服务发现命名的微服务应用程序。要使用此入门指南，您必须在已配置服务发现的 Amazon ECS 上运行微服务应用程序。

有关 Amazon ECS 上的服务发现的更多信息，请参阅服务发现 (p. 306)。

步骤 1：创建您的服务网格

服务网格是一种用于驻留在其内的服务之间的网络流量的逻辑边界。有关更多信息，请参阅 AWS App Mesh 用户指南 中的服务网格。

在创建服务网格后，您可以创建虚拟服务、虚拟节点、虚拟路由器以及用于在网格中的应用程序之间分配流量的路由。

使用 AWS 管理控制台创建新服务网格

1. 通过以下网址打开 App Mesh 控制台：https://console.aws.amazon.com/appmesh/。
2. 选择 Create mesh (创建网格)。
3. 对于 Mesh name (网格名称)，为您的服务网格指定一个名称。
4. 选择 Create mesh (创建网络) 以完成。

步骤 2：创建您的虚拟节点

虚拟节点充当某特定任务组的逻辑指针，如 Kubernetes 部署。有关更多信息，请参阅 AWS App Mesh 用户指南 中的虚拟节点。

当您创建虚拟节点时，必须为任务组指定服务发现方法。您的虚拟节点预计的任何入站流量均应指定为侦听器。您的虚拟节点预计达到的任何出站流量均应指定为后端。

您必须为应用程序中的每个微服务创建虚拟节点。

在 AWS 管理控制台 中创建虚拟节点。

1. 选择在先前步骤中创建的网格。

2. 在左侧导航中选择 Virtual nodes (虚拟节点)。
3. 选择 Create virtual node (创建虚拟节点)。
4. 对于 Virtual node name (虚拟节点名称)，为您的虚拟节点输入一个名称。
5. 对于 Service discovery method (服务发现方法)，选择以下选项之一：
   - DNS – 指定 DNS 主机名。
   - AWS Cloud Map – 指定服务名和命名空间。或者，您也可以指定 App Mesh 可以查询其属性的 AWS Cloud Map。只会返回匹配所有键/值对的实例。要使用 AWS Cloud Map，您的账户必须具有 `AWSServiceRoleForAppMesh`服务相关角色 (p. 392)。
6. 要为虚拟节点指定任何后端（对于出口流量）或配置入站和出站访问日志记录信息，请选择 Additional configuration (附加配置)
   a. 要指定后端，请选择 Add backend (添加后端) 并输入您的虚拟节点与之通信的虚拟服务的虚拟服务名称或完整 Amazon Resource Name (ARN)。重复执行此步骤，直到您的所有虚拟节点后端均包含在内。
   b. 要配置日志记录，请输入您希望 Envoy 使用的 HTTP 访问日志路径。建议您使用 `/dev/stdout` 路径，以便您可以使用 Docker 日志驱动程序来将 Envoy 日志导出至某个服务，如 Amazon CloudWatch Logs。

      Note

      日志必须仍由您应用程序中的代理进行接收并发送至目标。此文件路径仅指示 Envoy 要发送日志的位置。
7. 为 Listener (侦听器) 指定 Port (端口) 和 Protocol (协议)。
8. 如果要为您的侦听器配置运行状况检查，请确保已选中 Health check enabled (运行状况检查已启用)，然后完成以下子步骤。如果未选中，请清除此复选框。
   a. 对于 Health check protocol (运行状况检查协议)，选择使用 HTTP 或 TCP 运行状况检查。
   b. 对于 Health check port (运行状况检查端口)，指定应对其运行状况检查的端口。
   c. 对于 Healthy threshold (正常阈值)，指定在声明侦听器运行状况良好之前必须出现的连续成功的运行状况检查次数。
   d. 对于 Health check interval (运行状况检查间隔)，指定执行每次运行状况检查间隔的时间（毫秒）。
   e. 对于 Path (路径)，指定运行状况检查请求的目标路径。此参数仅在指定协议为 HTTP 时是必需的。如果协议是 TCP，将忽略此参数。
   f. 对于 Timeout period (超时周期)，指定接收来自运行状况检查的响应时要等待的时间（毫秒）。
   g. 对于 Unhealthy threshold (不正常阈值)，指定在声明侦听器运行状况不正常之前必须出现的连续失败的运行状况检查次数。
9. 选择 Create virtual node (创建虚拟节点) 以完成。
10. 必要时，重复此过程以便为您的应用程序中的每个剩余微服务创建虚拟节点。

步骤 3：创建您的虚拟路由器

虚拟路由器处理用于您的网格内一个或多个虚拟服务的流量。创建虚拟路由器后，您可以为您的虚拟路由器创建并关联路由，以将传入请求定向至不同的虚拟节点。 有关更多信息，请参阅 AWS App Mesh 用户指南中的虚拟路由器。

为您应用程序中的每个微服务创建虚拟路由器。

在 AWS 管理控制台 中创建虚拟路由器。

1. 在左侧导航中选择 Virtual routers (虚拟路由器)。
2. 选择 Create virtual router (创建虚拟路由器)。

3. 对于 Virtual router name (虚拟路由器名称)，为您的虚拟路由器指定一个名称。不得超过 255 个字母、数字、连字符和下划线的组合。
4. 对于 Listener (侦听器)，为您的虚拟路由器指定一个 Port (端口) 和 Protocol (协议)。
5. 选择 Create virtual router (创建虚拟路由器) 以完成。
6. 必要时，重复此过程以便为您的应用程序中的每个剩余微服务创建虚拟路由器。

步骤 4：创建您的路由

路由与虚拟路由器关联，用于匹配虚拟路由器的请求并将流量相应地分配到其关联的虚拟节点。 有关更多信息，请参阅 AWS App Mesh 用户指南 中的路由。

为您的应用程序中的每个微服务创建路由。

在 AWS 管理控制台 中创建路由。

1. 在左侧导航中选择 Virtual routers (虚拟路由器)。
2. 选择要将新路由与之关联的虚拟路由器。
3. 在 Routes (路由) 表中，选择 Create route (创建路由)。
4. 对于 Route name (路由名称)，指定要用于您的路由的名称。
5. 对于 Route type (路由类型)，选择您的路由的协议。
6. 对于 Virtual node name (虚拟节点名称)，选择此路由将流量提供给的虚拟节点。
7. 对于 Weight (权重)，为路由选择一个相对权重。选择 Add target (添加目标) 以添加其他虚拟节点。所有目标的总权重必须小于或等于 100。
8. 要使用基于 HTTP 路径的路由，选择 Additional configuration (附加配置)，然后指定该路由应匹配的路径。 例如，如果您的虚拟服务名称为 `service-b.local`，并且您希望路由将请求匹配到 `service-b.local/metrics`，则前缀应为 `/metrics`。
9. 选择 Create route (创建路由) 以完成。
10. 必要时，重复此过程以便为您的应用程序中的每个剩余微服务创建路由。

步骤 5：创建您的虚拟服务

虚拟服务是一种抽象的实际服务，由虚拟节点直接提供或通过虚拟路由器的方式间接提供。相关服务通过其 `virtualServiceName` 调用您的虚拟服务，然后这些请求将路由至指定为虚拟服务的提供商的虚拟节点或虚拟路由器。 有关更多信息，请参阅 AWS App Mesh 用户指南 中的虚拟服务。

为您的应用程序中的每个微服务创建虚拟服务。

在 AWS 管理控制台 中创建虚拟服务。

1. 在左侧导航中选择 Virtual services (虚拟服务)。
2. 选择 Create virtual service (创建虚拟服务)
3. 对于 Virtual service name (虚拟服务名称)，为您的虚拟服务选择一个名称。建议您使用要针对的实际服务的服务发现名称（如 `service-a.default.svc.cluster.local`）。
4. 对于 Provider (提供商)，选择您的虚拟服务的提供商类型：
   - 如果您希望虚拟服务跨多个虚拟节点分布流量，则选择 Virtual router (虚拟路由器)，然后从下拉菜单中选择要使用的虚拟路由器。
   - 如果您希望虚拟服务直接到达虚拟节点，而不使用虚拟路由器，则选择 Virtual node (虚拟节点)，然后从下拉菜单中选择要使用的虚拟节点。

- 如果您此时不希望虚拟服务路由流量（例如，如果您的虚拟节点或虚拟路由器不存在），则选择 None (无)。您可以稍后更新此虚拟服务的提供商。

5. 选择 Create virtual service (创建虚拟服务) 以完成。
6. 必要时，重复此过程以便为您的应用程序中的每个剩余微服务创建虚拟服务。

Update Your Microservice Task Definitions

Proxy Configuration

To configure your Amazon ECS service to use App Mesh, your service's task definition must have the following proxy configuration section. Set the proxy configuration `type` to `APPMESH` and the `containerName` to `envoy`. Set the following property values accordingly.

`IgnoredUID`

Envoy doesn't proxy traffic from processes that use this user ID. You can choose any user ID that you want for this (our examples use `1337` for historical purposes), but this ID must be the same as the `user` ID for the Envoy container in your task definition. This matching allows Envoy to ignore its own traffic without using the proxy.

`ProxyIngressPort`

This is the ingress port for the Envoy proxy container. Set this value to `15000`.

`ProxyEgressPort`

This is the egress port for the Envoy proxy container. Set this value to `15001`.

`AppPorts`

Specify any ingress ports that your application containers listen on. In this example, the application container listens on port `9080`.

`EgressIgnoredIPs`

Envoy doesn't proxy traffic to these IP addresses. Set this value to `169.254.170.2,169.254.169.254`, which ignores the Amazon EC2 metadata server and the Amazon ECS task metadata endpoint (which provides IAM roles for tasks credentials).

```
"proxyConfiguration": {
  "type": "APPMESH",
  "containerName": "envoy",
  "properties": [
    {
      "name": "IgnoredUID",
      "value": "1337"
    },
    {
      "name": "ProxyIngressPort",
      "value": "15000"
    },
    {
      "name": "ProxyEgressPort",
      "value": "15001"
    },
    {
      "name": "AppPorts",
      "value": "9080"
    },
```

```
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      }
    ]
  }
```

Application Container Envoy Dependency

The application containers in your task definitions must wait for the Envoy proxy to bootstrap and start before they can start. To ensure that this happens, you set a `dependsOn` section in each application container definition to wait for the Envoy container to report as `HEALTHY`. The following code block shows an application container definition example with this dependency.

```
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    }
```

Envoy Container Definition

Your Amazon ECS services' task definitions must contain the App Mesh custom Envoy container image.

```
111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-prod
```

The Envoy container definition must be marked as `essential`. The virtual node name for the Amazon ECS service should be set to the `APPMESH_VIRTUAL_NODE_NAME`, and the `user` ID value should match the `IgnoredUID` value from the task definition proxy configuration (in this example, we use `1337`). The health check shown here waits for the Envoy container to bootstrap properly before reporting to Amazon ECS that it is healthy and ready for the application containers to start.

The following code block shows an Envoy container definition example.

```
    {
      "name": "envoy",
      "image": "111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-
prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        }
      ],
```

```
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    }
```

Credentials

The Envoy container requires AWS Identity and Access Management credentials for signing requests that are sent to the App Mesh service. For Amazon ECS tasks deployed with the Amazon EC2 launch type, the credentials can come from the instance IAM role (p. 404) or from a task IAM role (p. 418). Amazon ECS tasks deployed with the Fargate launch type do not have access to the Amazon EC2 metadata server that supplies instance IAM profile credentials. To supply the credentials, you must attach an IAM task role to any tasks deployed with the Fargate launch type. The role doesn't need to have a policy attached to it, but for a task to work properly with App Mesh, the role must be attached to each task deployed with the Fargate launch type. If a task is deployed with the Amazon EC2 launch type and access is blocked to the Amazon EC2 metadata server, as described in the Important annotation in IAM Roles for Tasks (p. 418), then a task IAM role must also be attached to the task.

Update an Existing Task Definition

The Amazon ECS console assists in the process of updating your existing task definitions to add App Mesh integration.

Update a task definition to add App Mesh integration

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. From the navigation bar, choose the region that contains your task definition.
3. In the navigation pane, choose Task Definitions.
4. On the Task Definitions page, select the box to the left of the task definition to revise and choose Create new revision.
5. On the Create new revision of Task Definition page, make the following changes to enable App Mesh integration.

   a. For Service Integration, to configure the parameters for App Mesh integration choose Enable App Mesh integration and then do the following:

      i. For Application container name, choose the container name to use for the App Mesh application. This container must already be defined within the task definition.

      ii. For Envoy image, use the auto-populated Envoy container image which is 111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-prod.

      iii. For Mesh name, choose the App Mesh service mesh to use. This must already be created in order for it to show up. For more information, see Service Meshes in the AWS App Mesh User Guide.

      iv. For Virtual node name, choose the App Mesh virtual node to use. This must already be created in order for it to show up. For more information, see Virtual Nodes in the AWS App Mesh User Guide.

      v. For Virtual node port, this will be pre-populated with the listener port set on the virtual node.

vi. Choose Apply, Confirm. This will create a new Envoy proxy container to the task definition, as well as the settings to support it. It will then pre-populate the App Mesh proxy configuration settings for the next step.

b. For Proxy Configuration, verify all of the pre-populated values. For more information on these fields, see Proxy Configuration (p. 547).

6. Verify the information and choose Create.
7. If your task definition is used in a service, update your service with the updated task definition. For more information, see 更新服务 (p. 326).

Example Task Definitions

The following example Amazon ECS task definitions show, in context, the snippets that you can merge with your existing task groups. Substitute your mesh name and virtual node name for the `APPMESH_VIRTUAL_NODE_NAME` value and a list of ports that your application listens on for the proxy configuration `AppPorts` value.

If you're running an Amazon ECS task as described in the section called “Credentials” (p. 549), you need an existing task IAM role (p. 418). You should also add this line of code to the example task definitions that follow: `"taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole"`

Example JSON for Amazon ECS task definition

```
{
  "family": "appmesh-gateway",
  "memory": "256",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
      },
      {
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      }
    ]
  },
  "containerDefinitions": [
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
```

```
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-
prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Example JSON for Amazon ECS task definition with AWS X-Ray

X-Ray allows you to collect data about requests that an application serves and provides tools that you can use to visualize traffic flow. Using the X-Ray driver for Envoy enables Envoy to report tracing information to X-Ray. You can enable X-Ray tracing using the Envoy configuration. Based on the configuration, Envoy sends tracing data to the X-Ray daemon running as a sidecar container and the daemon forwards the traces to the X-Ray service. Once the traces are published to X-Ray, you can use the X-Ray console to visualize the service call graph and request trace details. The following JSON represents a task definition to enable X-Ray integration.

```
{
  "family": "appmesh-gateway",
  "memory": "256",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
      },
      {
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
```

```
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      }
    ]
  },
  "containerDefinitions": [
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.11.1.0-
prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
         "name": "ENABLE_ENVOY_XRAY_TRACING",
         "value": "1"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    },
    {
      "name": "xray-daemon",
      "image": "amazon/aws-xray-daemon",
      "user": "1337",
      "essential": true,
      "cpu": 32,
      "memoryReservation": 256,
```

```
      "portMappings": [
        {
          "containerPort": 2000,
          "protocol": "udp"
        }
      ]
    }
  ],
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Amazon ECS 上的 AWS Deep Learning Containers

AWS Deep Learning Containers 是一组 Docker 映像，用于在 Amazon Elastic Container Service (Amazon ECS) 上的 TensorFlow 中训练和处理模型。Deep Learning Containers 为优化环境提供了 TensorFlow、Nvidia CUDA（适用于 GPU 实例）和 Intel MKL（适用于 CPU 实例）库且在 Amazon ECR 中可用。

要开始使用 Amazon ECS 上的 Deep Learning Containers，请参阅 AWS Deep Learning AMI 开发人员指南中的 Amazon ECS 上的 Deep Learning Containers。

Amazon ECS 教程

以下教程说明在使用 Amazon ECS 时如何执行常见任务。

主题

- Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters (p. 555)
- 教程：使用 Secrets Manager 密钥指定敏感数据 (p. 557)
- 教程：使用蓝/绿部署创建服务 (p. 561)
- 教程：CodePipeline 持续部署 (p. 569)
- Tutorial: Using Amazon EFS File Systems with Amazon ECS (p. 573)

Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters

Container instances in your clusters need external network access to communicate with the Amazon ECS service endpoint. However, you might have tasks and services that you would like to run in private subnets. Creating a VPC with both public and private subnets provides you the flexibility to launch tasks and services in either a public or private subnet. Tasks and services in the private subnets can access the internet through a NAT gateway. Services in both the public and private subnets can be configured to use a load balancer so that they can still be reached from the public internet.

This tutorial guides you through creating a VPC with two public subnets and two private subnets, which are provided with internet access through a NAT gateway.

Step 1: Create an Elastic IP Address for Your NAT Gateway

A NAT gateway requires an Elastic IP address in your public subnet, but the VPC wizard does not create one for you. Create the Elastic IP address before running the VPC wizard.

To create an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose Elastic IPs.
3. Choose Allocate new address, Allocate, Close.
4. Note the Allocation ID for your newly created Elastic IP address; you enter this later in the VPC wizard.

Step 2: Run the VPC Wizard

The VPC wizard automatically creates and configures most of your VPC resources for you.

To run the VPC wizard

1. In the left navigation pane, choose VPC Dashboard.

2. Choose Start VPC Wizard, VPC with Public and Private Subnets, Select.
3. For VPC name, give your VPC a unique name.
4. For Elastic IP Allocation ID, choose the ID of the Elastic IP address that you created earlier.
5. Choose Create VPC.
6. When the wizard is finished, choose OK. Note the Availability Zone in which your VPC subnets were created. Your additional subnets should be created in a different Availability Zone.

Step 3: Create Additional Subnets

The wizard creates a VPC with a single public and a single private subnet in a single Availability Zone. For greater availability, you should create at least one more of each subnet type in a different Availability Zone so that your VPC has both public and private subnets across two Availability Zones.

To create an additional private subnet

1. In the left navigation pane, choose Subnets.
2. Choose Create Subnet.
3. For Name tag, enter a name for your subnet, such as Private subnet.
4. For VPC, choose the VPC that you created earlier.
5. For Availability Zone, choose a different Availability Zone than your original subnets in the VPC.
6. For IPv4 CIDR block, enter a valid CIDR block. For example, the wizard creates CIDR blocks in 10.0.0.0/24 and 10.0.1.0/24 by default. You could use 10.0.3.0/24 for your second private subnet.
7. Choose Yes, Create.

To create an additional public subnet

1. In the left navigation pane, choose Subnets and then Create Subnet.
2. For Name tag, enter a name for your subnet, such as Public subnet.
3. For VPC, choose the VPC that you created earlier.
4. For Availability Zone, choose the same Availability Zone as the additional private subnet that you created in the previous procedure.
5. For IPv4 CIDR block, enter a valid CIDR block. For example, the wizard creates CIDR blocks in 10.0.0.0/24 and 10.0.1.0/24 by default. You could use 10.0.2.0/24 for your second public subnet.
6. Choose Yes, Create.
7. Select the public subnet that you just created and choose Route Table, Edit.
8. By default, the private route table is selected. Choose the other available route table so that the 0.0.0.0/0 destination is routed to the internet gateway (igw-***xxxxxxxx***) and choose Save.
9. With your second public subnet still selected, choose Subnet Actions, Modify auto-assign IP settings.
10. Select Enable auto-assign public IPv4 address and choose Save, Close.

Next Steps

After you have created your VPC, you should consider the following next steps:

- Create security groups for your public and private resources if they require inbound network access. For more information, see Working with Security Groups in the Amazon VPC User Guide.
- Create Amazon ECS clusters in your private or public subnets. For more information, see 创建集群 (p. 38). If you use the cluster creation wizard in the Amazon ECS console, you can specify the VPC

that you just created and the public or private subnets in which to launch your instances, depending on your use case.
  - To make your containers directly accessible from the internet, launch instances into your public subnets. Be sure to configure your container instance security groups appropriately.
  - To avoid making containers directly accessible from the internet, launch instances into your private subnets.
- Create a load balancer in your public subnets that can route traffic to services in your public or private subnets. For more information, see 服务负载均衡 (p. 284).

教程：使用 Secrets Manager 密钥指定敏感数据

Amazon ECS 使您能够向容器中注入敏感数据，方法是将您的敏感数据存储在 AWS Secrets Manager 密钥中，然后在容器定义中引用它们。有关更多信息，请参阅指定敏感数据 (p. 107)。

以下教程说明如何创建 Secrets Manager 密钥，在 Amazon ECS 任务定义中引用该密钥，然后通过查询显示密钥内容的容器内的环境变量来验证它是否有效。

先决条件

本教程假设以下先决条件已完成：

- Amazon ECS 的设置 (p. 7) 中的步骤已完成。
- 您的 AWS 用户具有创建所述的 Secrets Manager 和 Amazon ECS 资源所需的 IAM 权限。

步骤 1：创建 Secrets Manager 密钥

您可以使用 Secrets Manager 控制台为您的敏感数据创建密钥。在本教程中，我们将创建一个基本密钥来存储用户名和密码以便稍后在容器中引用。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的创建基本密钥。

创建基本密钥

使用 Secrets Manager 为您的敏感数据创建密钥。

1. 通过 https://console.aws.amazon.com/secretsmanager/ 打开 Secrets Manager 控制台。
2. 选择存储新密钥。
3. 对于选择密钥类型，选择其他密钥类型。
4. 对于 Specify the key/value pairs to be stored in this secret (指定要存储在此密钥中的密钥/值对)，选择 Plaintext (纯文本) 选项卡，并将现有文本替换为以下文本。您在此处指定的文本值将是教程结尾处容器中的环境变量值。

   ```
   password_value
   ```

5. 选择 Next。
6. 对于 Secret name (密钥名称)，键入 `username_value`，然后选择 Next (下一步)。您在此处指定的密钥名称值将是教程结尾处容器中的环境变量名称。
7. 对于 Configure automatic rotation (配置自动轮换)，将 Disable automatic rotation (禁用自动轮换) 保持选中状态并选择 Next (下一步)。
8. 检查这些设置，然后选择 Store (存储) 以将输入的所有内容作为新密钥保存在 Secrets Manager 中。
9. 选择您刚刚创建的密钥，保存 Secret ARN (密钥 ARN) 以便在后续步骤中在任务执行 IAM 策略和任务定义中引用。

步骤 2：更新任务执行 IAM 角色

为了让 Amazon ECS 检索 Secrets Manager 密钥中的敏感数据，您必须具有 Amazon ECS 任务执行角色并在任务定义中引用它。这允许容器代理提取必要的 Secrets Manager 资源。如果您尚未创建任务执行 IAM 角色，请参阅 Amazon ECS 任务执行 IAM 角色 (p. 406)。

以下步骤假设您已创建并适当配置任务执行 IAM 角色。

更新任务执行 IAM 角色

使用 IAM 控制台更新具有所需权限的任务执行角色。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles。
3. 在角色列表中搜索 `ecsTaskExecutionRole` 并将其选定。
4. 选择 Permissions (权限)、Add inline policy (添加内联策略)。
5. 选择 JSON 选项卡并指定以下 JSON 文本，并确保指定您在步骤 1 中创建的 Secrets Manager 密钥的完整 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:region:aws_account_id:secret:username_value-
u9bH6K"
            ]
        }
    ]
}
```

6. 选择查看策略。对于 Name (名称)，指定 `ECSSecretsTutorial`，然后选择 Create policy (创建策略)。

步骤 3：创建 Amazon ECS 任务定义

您可以使用 Amazon ECS 控制台创建引用 Secrets Manager 密钥的任务定义。

创建指定密钥的任务定义

使用 IAM 控制台更新具有所需权限的任务执行角色。

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，依次选择 Task Definitions (任务定义) 和 Create new Task Definition (创建新任务定义)。
3. 在 Select launch type compatibility (选择启动类型兼容性) 页面上，选择 EC2，然后选择 Next step (下一步)。
4. 选择 Configure via JSON (通过 JSON 配置) 并输入以下任务定义 JSON 文本，确保指定您在步骤 1 中创建的 Secrets Manager 密钥的完整 ARN 以及您在步骤 2 中更新的任务执行 IAM 角色。选择 Save (保存)。

Important

任务定义中的密钥名称的值必须与您在创建密钥时为密钥名称指定的名称匹配。

```
{
    "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
            ],
            "cpu": 10,
            "secrets": [
                {
                    "valueFrom":
 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value-u9bH6K",
                    "name": "username_value"
                }
            ],
            "memory": 300,
            "image": "httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

5. 检查设置，然后选择 Create (创建)。

步骤 4：创建 Amazon ECS 集群

您可以使用 Amazon ECS 控制台创建一个包含要运行任务的容器实例的集群。如果现有集群中至少注册了一个容器实例，并且具有可用资源来运行为本教程创建的任务定义的一个实例，则可跳到下一步。

在本教程中，我们将使用 Amazon ECS-optimized Amazon Linux 2 AMI 创建具有一个 `t2.micro` 容器实例的集群。

创建群集

使用 Amazon ECS 控制台创建一个集群并向该集群注册一个容器实例。

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 从导航栏中，选择同时包含 Secrets Manager 密钥和您创建的 Amazon ECS 任务定义的区域。
3. 在导航窗格中，选择 Clusters。

4. 在 Clusters 页面上，选择 Create Cluster。
5. 对于 EC2 实例类型，选择 t2.micro。
6. 对于密钥对，选择要添加到容器实例的密钥对。

   Important

   需要密钥对才能完成本教程，因此，如果您还没有创建密钥对，请转至指向 EC2 控制台的链接来创建密钥对。
7. 将所有其他字段保留为默认值，然后选择创建。

步骤 5：运行 Amazon ECS 任务

您可以使用 Amazon ECS 控制台通过创建的任务定义运行任务。在本教程中，我们将使用上一步中创建的集群来运行使用 EC2 启动类型的任务。

运行任务

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Task Definitions (任务定义)，然后选择我们创建的 ecs-secrets-tutorial 任务定义。
3. 选择最新修订版的任务定义，然后选择 Actions (操作)、Run Task (运行任务)。
4. 对于启动类型，选择 EC2。
5. 对于 Cluster (集群)，选择我们在上一步中创建的 ecs-secrets-tutorial 集群。
6. 对于 Task tagging configuration (任务标记配置)，取消选择 Enable ECS managed tags (启用 ECS 托管标签)。对于本教程而言，它们是不必要的。
7. 查看您的任务信息并选择 Run Task。

   Note

   如果您的任务从 `PENDING` 移至 `STOPPED`，或者它显示 `PENDING` 状态，然后从列出的任务中消失，则您的任务可能因出错而停止。有关更多信息，请参阅故障排除部分中的检查已停止的任务是否存在错误 (p. 579)。

步骤 6：验证

您可以使用以下步骤验证是否已成功完成所有步骤并在容器中正确创建了环境变量。

验证是否已创建环境变量

1. 查找您的容器实例的公共 IP 或 DNS 地址。
   a. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
   b. 选择托管您的容器实例的 `ecs-secrets-tutorial` 集群。
   c. 在 Cluster (集群) 页面上，选择 ECS Instances (ECS 实例)。
   d. 在 Container Instance 列上，选择要连接到的容器实例。
   e. 在 Container Instance 页面上，记录您的实例的 Public IP 或 Public DNS。
2. 如果您使用的是 macOS 或 Linux 计算机，请使用以下命令连接到您的实例，并替换您的私有密钥的路径和实例的公共地址：

   ```
   $ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
   ```

   有关使用 Windows 计算机的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的使用 PuTTY 从 Windows 连接到您的 Linux 实例。

> **Important**
>
> 有关在连接到实例时遇到的任何问题的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的排查实例的连接问题。

3. 列出正在实例上运行的容器。记下 `ecs-secrets-tutorial` 容器的容器 ID。

```
docker ps
```

4. 使用上一步的输出中的容器 ID 连接到 `ecs-secrets-tutorial` 容器。

```
docker exec -it container_ID /bin/bash
```

5. 使用 `echo` 命令打印环境变量的值。

```
echo $username_value
```

如果教程成功完成，您应看到以下输出：

```
password_value
```

> **Note**
>
> 或者，您可以使用 `env`（或 `printenv`）命令列出容器中的所有环境变量。

步骤 7：清除

完成本教程后，您应清除相关资源，以避免产生与未使用的资源相关的费用。

清除资源

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 选择您创建的 ecs-secrets-tutorial cluster (ecs-secrets-tutorial 集群)。
3. 在 Cluster (集群) 页面上，选择 Delete Cluster (删除集群)。
4. 输入删除集群确认短语，然后选择 Delete (删除)。这将需要几分钟的时间，但将清除所有 Amazon ECS 集群资源。
5. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
6. 在导航窗格中，选择 Roles。
7. 在角色列表中搜索 `ecsTaskExecutionRole` 并将其选定。
8. 选择权限，然后选择 `ECSSecretsTutorial` 旁边的 X。选择删除以确认删除内联策略。
9. 通过 https://console.aws.amazon.com/secretsmanager/ 打开 Secrets Manager 控制台。
10. 选择您之前创建的 username_value 密钥，然后选择操作、删除密钥。

教程：使用蓝/绿部署创建服务

Amazon ECS 已将蓝/绿部署集成到 Amazon ECS 控制台的创建服务向导中。有关更多信息，请参阅创建服务 (p. 317)。

以下教程说明如何创建一个 Amazon ECS 服务，该服务包含一个将蓝/绿部署类型与 AWS CLI 结合使用的 Fargate 任务。

先决条件

本教程假设您已完成以下先决条件：

- 安装并配置了最新版本的 AWS CLI。有关安装或升级 AWS CLI 的更多信息，请参阅安装 AWS Command Line Interface。
- Amazon ECS 的设置 (p. 7) 中的步骤已完成。
- 您的 AWS 用户具有 Amazon ECS 首次运行向导权限 (p. 376) IAM 策略示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。有关更多信息，请参阅教程：为您的集群创建带有公有和私有子网的 VPC。

第 1 步：创建应用程序负载均衡器

使用蓝/绿部署类型的 Amazon ECS 服务需要使用 应用程序负载均衡器 或 网络负载均衡器。本教程使用 应用程序负载均衡器。

创建 应用程序负载均衡器

1. 使用 create-load-balancer 命令创建 应用程序负载均衡器。指定两个不属于同一可用区的子网以及一个安全组。

```
aws elbv2 create-load-balancer --name bluegreen-alb \
--subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --region us-
east-1
```

   输出包含负载均衡器的 Amazon 资源名称 (ARN)，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. 使用 create-target-group 命令创建目标组。此目标组将流量路由到服务中的原始任务集。

```
aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

   输出包含目标组的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. 使用 create-listener 命令创建负载均衡器侦听器，该侦听器带有将请求转发到目标组的默认规则。

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

   输出包含侦听器的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

步骤 2：创建 Amazon ECS 集群

使用 create-cluster 命令创建要使用的名为 `tutorial-bluegreen` 的集群。

```
aws ecs create-cluster --cluster-name tutorial-bluegreen --region us-east-1
```

输出包含集群的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:cluster/tutorial-bluegreen
```

第 3 步：注册任务定义

使用 register-task-definition 命令注册与 Fargate 兼容的任务定义。它需要使用 `awsvpc` 网络模式。以下为用于本教程的示例任务定义。

首先，使用以下内容创建名为 `fargate-task.json` 的文件。确保您使用的是任务执行角色的 ARN。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!
</h2> <p>Your application is now running on a container in Amazon ECS.</p> </div></body></
html>' >  /usr/local/apache2/htdocs/index.html && httpd-foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512",
        "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole"
}
```

然后，使用您创建的 `fargate-task.json` 文件注册任务定义。

```
aws ecs register-task-definition --cli-input-json file://fargate-task.json --region us-
east-1
```

步骤 4：创建 Amazon ECS 服务

使用 create-service 命令创建服务。

首先，使用以下内容创建名为 `service-bluegreen.json` 的文件。

```
{
    "cluster": "tutorial-bluegreen",
    "serviceName": "service-bluegreen",
    "taskDefinition": "tutorial-task-def",
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4",
            "containerName": "sample-app",
            "containerPort": 80
        }
    ],
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234", "subnet-abcd5678" ]
       }
    },
    "desiredCount": 1
}
```

然后，使用您创建的 `service-bluegreen.json` 文件创建服务。

```
aws ecs create-service --cli-input-json file://service-bluegreen.json --region us-east-1
```

输出包含该服务的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:service/service-bluegreen
```

步骤 5：创建 AWS CodeDeploy 资源

使用以下步骤创建 CodeDeploy 应用程序、CodeDeploy 部署组的 应用程序负载均衡器 目标组和 CodeDeploy 部署组。

创建 CodeDeploy 资源

1. 使用 create-application 命令创建 CodeDeploy 应用程序。指定 `ECS` 计算平台。

   ```
   aws deploy create-application --application-name tutorial-bluegreen \
   --compute-platform ECS --region us-east-1
   ```

   输出包含应用程序 ID，格式如下：

   ```
   {
   ```

```
    "applicationId": "b8e9c1ef-3048-424e-9174-885d7dc9dc11"
}
```

2. 使用 create-target-group 命令创建另一个 应用程序负载均衡器 目标组（在创建 CodeDeploy 部署组时将使用该目标组）。

```
aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id "vpc-0b6dd82c67d8012a1" --region us-east-1
```

输出包含目标组的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc
```

3. 使用 create-deployment-group 命令创建 CodeDeploy 部署组。

首先，使用以下内容创建名为 `tutorial-deployment-group.json` 的文件。此示例使用您创建的资源。

```
{
   "applicationName": "tutorial-bluegreen",
   "autoRollbackConfiguration": {
      "enabled": true,
      "events": [ "DEPLOYMENT_FAILURE" ]
   },
   "blueGreenDeploymentConfiguration": {
      "deploymentReadyOption": {
         "actionOnTimeout": "CONTINUE_DEPLOYMENT",
         "waitTimeInMinutes": 0
      },
      "terminateBlueInstancesOnDeploymentSuccess": {
         "action": "TERMINATE",
         "terminationWaitTimeInMinutes": 5
      }
   },
   "deploymentGroupName": "tutorial-bluegreen-dg",
   "deploymentStyle": {
      "deploymentOption": "WITH_TRAFFIC_CONTROL",
      "deploymentType": "BLUE_GREEN"
   },
   "loadBalancerInfo": {
      "targetGroupPairInfoList": [
        {
          "targetGroups": [
             {
                 "name": "bluegreentarget1"
             },
             {
                 "name": "bluegreentarget2"
             }
          ],
          "prodTrafficRoute": {
              "listenerArns": [
                  "arn:aws:elasticloadbalancing:region:aws_account_id:listener/
app/bluegreen-alb/e5ba62739c16e642/665750bec1b03bd4"
              ]
          }
        }
      ]
   },
   "serviceRoleArn": "arn:aws:iam::aws_account_id:role/ecsCodeDeployRole",
   "ecsServices": [
       {
```

```
            "serviceName": "service-bluegreen",
            "clusterName": "tutorial-bluegreen"
        }
    ]
}
```

然后创建 CodeDeploy 部署组。

```
aws deploy create-deployment-group --cli-input-json file://tutorial-deployment-group.json --region us-east-1
```

输出包含部署组 ID，格式如下：

```
{
    "deploymentGroupId": "6fd9bdc6-dc51-4af5-ba5a-0a4a72431c88"
}
```

步骤 5：创建和监控 CodeDeploy 部署

使用以下步骤创建和上传应用程序规范文件（AppSpec 文件）和 CodeDeploy 部署。

创建和监控 CodeDeploy 部署

1. 使用以下步骤创建和上传 AppSpec 文件。

   a. 使用 CodeDeploy 部署组的内容创建名为 `appspec.yaml` 的文件。此示例使用您在本教程前面创建的资源。

   ```
   version: 0.0
   Resources:
     - TargetService:
         Type: AWS::ECS::Service
         Properties:
           TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-definition/first-run-task-definition:7"
           LoadBalancerInfo:
             ContainerName: "sample-app"
             ContainerPort: 80
           PlatformVersion: "LATEST"
   ```

   b. 使用 s3 mb 命令为 AppSpec 文件创建 Amazon S3 存储桶。

   ```
   aws s3 mb s3://tutorial-bluegreen
   ```

   c. 使用 s3 cp 命令将 AppSpec 文件上传到 Amazon S3 存储桶。

   ```
   aws s3 cp ./AppSpec.yaml s3://tutorial-bluegreen/appspec.yaml
   ```

2. 使用以下步骤创建 CodeDeploy 部署。

   a. 使用 CodeDeploy 部署的内容创建名为 `create-deployment.json` 的文件。此示例使用您在本教程前面创建的资源。

   ```
   {
       "applicationName": "tutorial-bluegreen",
       "deploymentGroupName": "tutorial-bluegreen-dg",
       "revision": {
   ```

```
        "revisionType": "S3",
        "s3Location": {
            "bucket": "tutorial-bluegreen",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

b. 使用 create-deployment 命令创建部署。

```
aws deploy create-deployment --cli-input-json file://create-deployment.json --
region us-east-1
```

输出包含部署 ID，格式如下：

```
{
    "deploymentId": "d-RPCR1U3TW"
}
```

c. 使用 get-deployment-target 命令获取部署的详细信息，并指定上一输出中的 deploymentId。

```
aws deploy get-deployment-target --deployment-id "d-IMJU3A8TW" --target-
id tutorial-bluegreen:service-bluegreen --region us-east-1
```

继续检索部署详细信息，直到状态为 Succeeded，如以下输出所示。

```
{
    "deploymentTarget": {
        "deploymentTargetType": "ECSTarget",
        "ecsTarget": {
            "deploymentId": "d-RPCR1U3TW",
            "targetId": "tutorial-bluegreen:service-bluegreen",
            "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-
bluegreen",
            "lastUpdatedAt": 1543431490.226,
            "lifecycleEvents": [
                {
                    "lifecycleEventName": "BeforeInstall",
                    "startTime": 1543431361.022,
                    "endTime": 1543431361.433,
                    "status": "Succeeded"
                },
                {
                    "lifecycleEventName": "Install",
                    "startTime": 1543431361.678,
                    "endTime": 1543431485.275,
                    "status": "Succeeded"
                },
                {
                    "lifecycleEventName": "AfterInstall",
                    "startTime": 1543431485.52,
                    "endTime": 1543431486.033,
                    "status": "Succeeded"
                },
                {
                    "lifecycleEventName": "BeforeAllowTraffic",
                    "startTime": 1543431486.838,
                    "endTime": 1543431487.483,
                    "status": "Succeeded"
                },
                {
```

```
                    "lifecycleEventName": "AllowTraffic",
                    "startTime": 1543431487.748,
                    "endTime": 1543431488.488,
                    "status": "Succeeded"
                },
                {
                    "lifecycleEventName": "AfterAllowTraffic",
                    "startTime": 1543431489.152,
                    "endTime": 1543431489.885,
                    "status": "Succeeded"
                }
            ],
            "status": "Succeeded",
            "taskSetsInfo": [
                {
                    "identifer": "ecs-svc/9223370493425779968",
                    "desiredCount": 1,
                    "pendingCount": 0,
                    "runningCount": 1,
                    "status": "ACTIVE",
                    "trafficWeight": 0.0,
                    "targetGroup": {
                        "name": "bluegreentarget1"
                    }
                },
                {
                    "identifer": "ecs-svc/9223370493423413672",
                    "desiredCount": 1,
                    "pendingCount": 0,
                    "runningCount": 1,
                    "status": "PRIMARY",
                    "trafficWeight": 100.0,
                    "targetGroup": {
                        "name": "bluegreentarget2"
                    }
                }
            ]
        }
    }
}
```

步骤 6：清除

完成本教程后，请清除与其关联的资源，以避免产生与您未使用的资源相关的费用。

清除教程资源

1. 使用 delete-deployment-group 命令删除 CodeDeploy 部署组。

```
aws deploy delete-deployment-group --application-name tutorial-bluegreen --deployment-group-name tutorial-bluegreen-dg --region us-east-1
```

2. 使用 delete-application 命令删除 CodeDeploy 应用程序。

```
aws deploy delete-application --application-name tutorial-bluegreen --region us-east-1
```

3. 使用 delete-service 命令删除 Amazon ECS 服务。通过使用 `--force` 标记，您可以删除服务，即使它尚未缩减至 0 个任务。

```
aws ecs delete-service --service arn:aws:ecs:region:aws_account_id:service/service-
bluegreen --force --region us-east-1
```

4. 使用 delete-cluster 命令删除 Amazon ECS 集群。

```
aws ecs delete-cluster --cluster tutorial-bluegreen --region us-east-1
```

5. 使用 s3 rm 命令从 Amazon S3 存储桶中删除 AppSpec 文件。

```
aws s3 rm s3://tutorial-bluegreen/appspec.yaml
```

6. 使用 s3 rb 命令删除 Amazon S3 存储桶。

```
aws s3 rb s3://tutorial-bluegreen
```

7. 使用 delete-load-balancer 命令删除 应用程序负载均衡器。

```
aws elbv2 delete-load-balancer --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 --region us-east-1
```

8. 使用 delete-target-group 命令删除两个 应用程序负载均衡器 目标组。

```
aws elbv2 delete-target-group --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

```
aws elbv2 delete-target-group --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc --region us-east-1
```

教程：CodePipeline 持续部署

本教程帮助您通过将 Amazon ECS 与 CodePipeline 结合使用来创建一个完整的端到端持续部署 (CD) 管道。

先决条件

您必须先部署一些资源，然后才能使用本教程创建您的 CD 管道。以下是您在开始操作之前需要的资源：

> **Note**
>
> 所有这些资源均应在同一 AWS 区域内创建。

- 一个包含您的 Dockerfile 和应用程序源的源代码控制存储库 (本教程使用 CodeCommit)。有关更多信息，请参阅 AWS CodeCommit 用户指南 中的创建 CodeCommit 存储库。
- 一个 Docker 映像存储库 (本教程使用 Amazon ECR)，其包含您从 Dockerfile 和应用程序源构建的映像。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南 中的创建存储库和推送映像。
- 一个 Amazon ECS 任务定义，该定义引用了在您的映像存储库中托管的 Docker 映像。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的创建任务定义。
- 一个 Amazon ECS 集群，该集群运行了使用您的前述任务定义的服务。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的创建集群和创建服务。

在满足这些先决条件后，您可以继续完成本教程并创建您的 CD 管道。

步骤 1：将构建规范文件添加至您的源存储库

本教程使用 CodeBuild 构建您的 Docker 映像并将该映像推送至 Amazon ECR。将 `buildspec.yml` 文件添加至您的源代码存储库以指示 CodeBuild 如何执行此操作。下面的示例构建规范将执行以下操作：

- 构建前阶段：
  - 登录 Amazon ECR。
  - 将存储库 URI 设置为您的 ECR 映像并添加包含源的 Git 提交 ID 的前七个字符的映像标签。
- 构建阶段：
  - 构建 Docker 映像并使用 Git 提交 ID 将该映像标记为 `latest`。
- 构建后阶段：
  - 使用两个标签将该映像推送至 ECR 存储库。
  - 将名为 `imagedefinitions.json` 的文件写入到包含您的 Amazon ECS 服务的容器名称以及该映像和标签的构建根目录。您的 CD 管道的部署阶段将使用此信息来创建您的服务的任务定义的新修订，然后它会将该服务更新为使用此新任务定义。CodeDeploy ECS 作业辅助角色需要 `imagedefinitions.json` 文件。

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"hello-world","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG >
 imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```

针对以下任务定义编写了构建规范，该规范由本教程的 Amazon ECS 服务使用。`REPOSITORY_URI` 值对应于 `image` 存储库 (没有任何映像标签)，此文件末尾附近的 *`hello-world`* 值对应于该服务的任务定义中的容器名称。

```
{
  "taskDefinition": {
    "family": "hello-world",
    "containerDefinitions": [
      {
        "name": "hello-world",
```

```
        "image": "012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world:6a57b99",
        "cpu": 100,
        "portMappings": [
          {
            "protocol": "tcp",
            "containerPort": 80,
            "hostPort": 80
          }
        ],
        "memory": 128,
        "essential": true
      }
    ]
  }
}
```

将 **buildspec.yml** 文件添加至您的源存储库

1. 打开文本编辑器，然后将上述构建规范复制并粘贴到新文件中。
2. 针对您的 Docker 映像将 `REPOSITORY_URI` 值 (*012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world*) 替换为 Amazon ECR 存储库 URI（没有任何映像标签）。将 *hello-world* 替换为引用您的 Docker 映像的服务的任务定义中的容器名称。
3. 提交您的 `buildspec.yml` 文件并将其推送至源存储库。

   a. 添加文件。

   ```
   git add .
   ```

   b. 提交更改。

   ```
   git commit -m "Adding build specification."
   ```

   c. 推送提交。

   ```
   git push
   ```

步骤 2：创建持续部署管道

使用 CodePipeline 向导创建您的管道阶段并将源存储库连接到您的 ECS 服务。

创建管道

1. 通过以下网址打开 CodePipeline 控制台：https://console.aws.amazon.com/codepipeline/。
2. 在 Welcome 页面上，选择 Create pipeline。

   如果这是您第一次使用 CodePipeline，则会显示一个介绍页面，而不是欢迎页面。选择 Get Started Now。
3. 在 Step 1: Name 页面上，对于 Pipeline name，键入您的管道的名称，然后选择 Next step。在本教程中，管道名称为 hello-world。
4. 在 步骤 2: 源 页面上，对于源提供商，选择 CodeCommit。

   a. 对于存储库名称，选择要用作您的管道的源位置的 CodeCommit 存储库的名称。
   b. 对于 Branch name，选择要使用的分支，然后选择 Next step。
5. 在步骤 3: 构建页面上，选择 CodeBuild，然后选择创建新构建项目。

   a. 对于 Project name，为您的构建项目选择唯一名称。在本教程中，项目名称为 hello-world。

b. 对于 Operating system，选择 Ubuntu。
c. 对于 Runtime，选择 Docker。
d. 对于版本，选择 aws/codebuild/docker:17.09.0。
e. 选择 Save build project。
f. 选择下一步。

> **Note**
>
> 该向导为您的构建项目创建一个名为 code-build-***build-project-name***-service-role 的 CodeBuild 服务角色。记下此角色名称，因为稍后您要将 Amazon ECR 权限添加至该角色。

6. 在 步骤 4: 部署页面上，对于部署提供商，选择 Amazon ECS。
   a. 对于集群名称，选择您的服务正在其中运行的 Amazon ECS 集群。在本教程中，该集群为 default。
   b. 对于 Service name，选择要更新的服务，然后选择 Next step。在本教程中，服务名称为 hello-world。
7. 在 Step 5: Service Role 页面上，选择 Create role。在介绍要为您创建的角色的 IAM 控制台页面上，选择允许。
8. 选择下一步。
9. 在 Step 6: Review 页面上，审查您的管道配置，然后选择 Create pipeline 以创建管道。

> **Note**
>
> 既然管道已创建好，它将尝试经历不同的管道阶段。但是，由此向导创建的默认 CodeBuild 角色无权执行包含在 `buildspec.yml` 文件中的所有命令，因此构建阶段失败。下一部分将为构建阶段添加权限。

步骤 3：向 CodeBuild 角色添加 Amazon ECR 权限

CodePipeline 向导为 CodeBuild 构建项目创建了一个名为 code-build-***build-project-name***-service-role 的 IAM 角色。在本教程中，该名称为 code-build-hello-world-service-role。由于 `buildspec.yml` 文件调用了 Amazon ECR API 操作，因此该角色必须具有一个允许权限进行这些 Amazon ECR 调用的策略。以下过程帮助您将适当权限附加到该角色。

向 CodeBuild 角色添加 Amazon ECR 权限

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles。
3. 在搜索框中，键入 code-build-，然后选择已由 CodePipeline 向导创建的角色。在本教程中，角色名称为 code-build-hello-world-service-role。
4. 在 Summary 页面上，选择 Attach policy。
5. 选中 AmazonEC2ContainerRegistryPowerUser 策略左侧的框，然后选择 Attach policy。

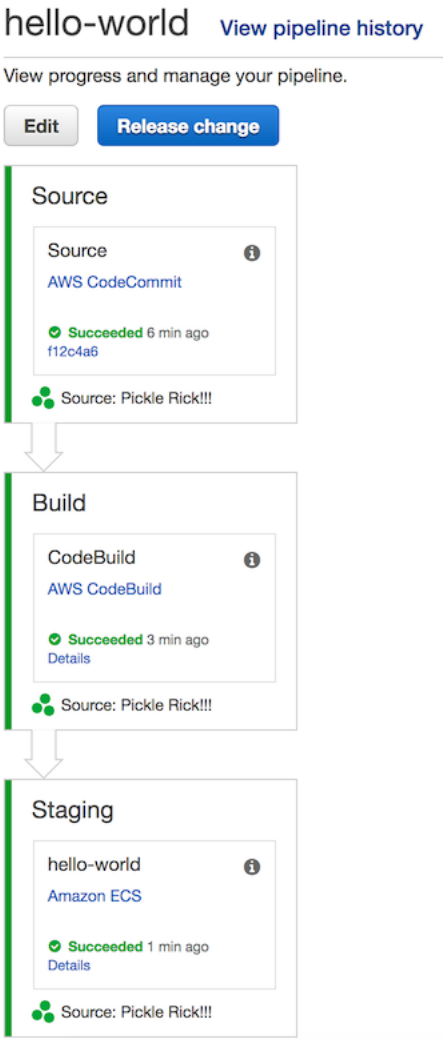
步骤 4：测试您的管道

您的管道应具有运行端到端本机 AWS 持续部署所需的一切。现在，通过将代码更改推送至您的源存储库来测试管道的功能。

测试您的管道

1. 对您的已配置源存储库进行代码更改，然后提交并推送更改。

2. 通过以下网址打开 CodePipeline 控制台：https://console.aws.amazon.com/codepipeline/。
3. 从列表中选择您的管道。
4. 监视管道经历不同阶段的进度。您的管道应完成且您的 Amazon ECS 服务运行通过您的代码更改创建的 Docker 映像。



Tutorial: Using Amazon EFS File Systems with Amazon ECS

Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files. Your applications can have the storage they need, when they need it.

You can use Amazon EFS file systems with Amazon ECS to export file system data across your fleet of container instances. That way, your tasks have access to the same persistent storage, no matter the instance on which they land. However, you must configure your container instance AMI to mount the Amazon EFS file system before the Docker daemon starts. Also, your task definitions must reference volume mounts on the container instance to use the file system. The following sections help you get started using Amazon EFS with Amazon ECS.

Note

Amazon EFS is not available in all regions. For more information about which regions support Amazon EFS, see Amazon Elastic File System in the AWS Regions and Endpoints section of the AWS General Reference.

Step 1: Gather Cluster Information

Before you can create all of the required resources to use Amazon EFS with your Amazon ECS cluster, gather some basic information about the cluster, such as the VPC it is hosted inside of, and the security group that it uses.

To gather the VPC and security group IDs for a cluster

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Select one of the container instances from your cluster and view the Description tab of the instance details. If you created your cluster with the Amazon ECS first-run or cluster creation wizards, the cluster name should be part of the EC2 instance name. For example, a cluster named `default` has this EC2 instance name: `ECS Instance - EC2ContainerService-`*`default`*.
3. Record the VPC ID value for your container instance. Later, you create a security group and an Amazon EFS file system in this VPC.
4. Open the security group to view its details.
5. Record the Group ID. Later, you allow inbound traffic from this security group to your Amazon EFS file system.

Step 2: Create a Security Group for an Amazon EFS File System

In this section, you create a security group for your Amazon EFS file system that allows inbound access from your container instances.

To create a security group for an Amazon EFS file system

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the left navigation pane, choose Security Groups, Create Security Group.
3. For Security group name, enter a unique name for your security group. For example, `EFS-access-for-sg-`*`dc025fa2`*.
4. For Description, enter a description for your security group.
5. For VPC, choose the VPC that you identified earlier for your cluster.
6. Choose Inbound, Add rule.
7. For Type, choose NFS.
8. For Source, choose Custom and then enter the security group ID that you identified earlier for your cluster.
9. Choose Create.

Step 3: Create an Amazon EFS File System

Before you can use Amazon EFS with your container instances, you must create an Amazon EFS file system.

To create an Amazon EFS file system for Amazon ECS container instances

1. Open the Amazon Elastic File System console at https://console.aws.amazon.com/efs/.

   Note

   Amazon EFS is not available in all regions. For more information about which regions support Amazon EFS, see Amazon Elastic File System in the AWS Regions and Endpoints section of the AWS General Reference.

2. Choose Create file system.
3. On the Configure file system access page, choose the VPC that your container instances are hosted in. By default, each subnet in the specified VPC receives a mount target that uses the default security group for that VPC.

   Note

   Your Amazon EFS file system and your container instances must be in the same VPC.

4. Under Create mount targets, for Security groups, add the security group that you created in the previous section. Choose Next step.
5. Configure optional settings and then choose Next step to proceed.

   a. (Optional) Add tags for your file system. For example, you could specify a unique name for the file system by entering that name in the Value column next to the Name key.
   b. (Optional) Enable lifecycle management to save money on infrequently accessed storage. For more information, see EFS Lifecycle Management in the Amazon Elastic File System User Guide.
   c. Choose a throughput mode for your file system.

      Note

      Bursting is the default, and it is recommended for most file systems.

   d. Choose a performance mode for your file system.

      Note

      General Purpose is the default, and it is recommended for most file systems.

   e. (Optional) Enable encryption. Select the check box to enable encryption of your Amazon EFS file system at rest.

6. Review your file system options and choose Create File System.

Step 4: Configure Container Instances

After you've created your Amazon EFS file system in the same VPC as your container instances, you must configure the container instances to access and use the file system.

Configure a running container instance to use an Amazon EFS file system

1. Log in to the container instance via SSH. For more information, see 连接到您的容器实例 (p. 180).
2. Create a mount point for your Amazon EFS file system. For example, `/efs`.

```
sudo mkdir /mnt/efs
```

3. Install the `amazon-efs-utils` client software on your container instance.

   - For Amazon Linux:

```
sudo yum install -y amazon-efs-utils
```

- For other Linux distributions, see Installing the amazon-efs-utils Package on Other Linux Distributions in the Amazon Elastic File System User Guide.

4. Make a backup of the `/etc/fstab` file.

```
sudo cp /etc/fstab /etc/fstab.bak
```

5. Update the `/etc/fstab` file to automatically mount the file system at boot.

```
echo 'fs-12345678:/ /mnt/efs efs defaults,_netdev 0 0' | sudo tee -a /etc/fstab
```

6. Reload the file system table to verify that your mounts are working properly.

```
sudo mount -a
```

> **Note**
>
> If you receive an error while running the above command, examine your `/etc/fstab` file for problems. If necessary, restore it with the backup that you created earlier.

7. Validate that the file system is mounted correctly with the following command. You should see a file system entry that matches your Amazon EFS file system. If not, see Troubleshooting Amazon EFS in the Amazon Elastic File System User Guide.

```
mount | grep efs
```

Bootstrap an instance to use Amazon EFS with user data

You can use an Amazon EC2 user data script to bootstrap an Amazon ECS–optimized AMI at boot. For more information, see 使用 Amazon EC2 用户数据引导启动容器实例 (p. 168).

1. Follow the container instance launch instructions at 启动 Amazon ECS 容器实例 (p. 165).
2. On Step 6.g (p. 166), pass the following user data to configure your instance. If you are not using the `default` cluster, be sure to replace the `ECS_CLUSTER=default` line in the configuration file to specify your own cluster name.

```
Content-Type: multipart/mixed; boundary="==BOUNDARY==="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/cloud-boothook; charset="us-ascii"

# Install amazon-efs-utils
cloud-init-per once yum_update yum update -y
cloud-init-per once install_amazon-efs-utils yum install -y amazon-efs-utils

# Create /efs folder
cloud-init-per once mkdir_efs mkdir /efs

# Mount /efs
cloud-init-per once mount_efs echo -e 'fs-12345678:/ /efs efs defaults,_netdev 0 0'
 >> /etc/fstab
mount -a

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
```

```
#!/bin/bash
# Set any ECS agent configuration options
echo "ECS_CLUSTER=default" >> /etc/ecs/ecs.config

--==BOUNDARY==--
```

Step 5: Create a Task Definition to Use the Amazon EFS File System

Because the file system is mounted on the host container instance, you must create a volume mount in your Amazon ECS task definition that allows your containers to access the file system. For more information, see 在任务中使用数据卷 (p. 83).

The following task definition creates a data volume called `efs-html` at `/efs/html` on the host container instance Amazon EFS file system. The `nginx` container mounts the host data volume at the NGINX root, `/usr/share/nginx/html`.

```
{
  "containerDefinitions": [
    {
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "containerPath": "/usr/share/nginx/html",
          "sourceVolume": "efs-html"
        }
      ],
      "name": "nginx",
      "image": "nginx"
    }
  ],
  "volumes": [
    {
      "host": {
        "sourcePath": "/efs/html"
      },
      "name": "efs-html"
    }
  ],
  "family": "nginx-efs"
}
```

You can save this task definition to a file called `nginx-efs.json` and register it to use in your own clusters with the following AWS CLI command. For more information, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.

```
aws ecs register-task-definition --cli-input-json file://nginx-efs.json
```

Step 6: Add Content to the Amazon EFS File System

For the NGINX example task, you created a directory at `/efs/html` on the container instance to host the web content. Before the NGINX containers can serve any web content, you must add the content to the file system. In this section, you log in to a container instance and add an `index.html` file.

To add content to the file system

1. Connect using SSH to one of your container instances that is using the Amazon EFS file system. For more information, see 连接到您的容器实例 (p. 180).
2. Write a simple HTML file by copying and pasting the following block of text into a terminal.

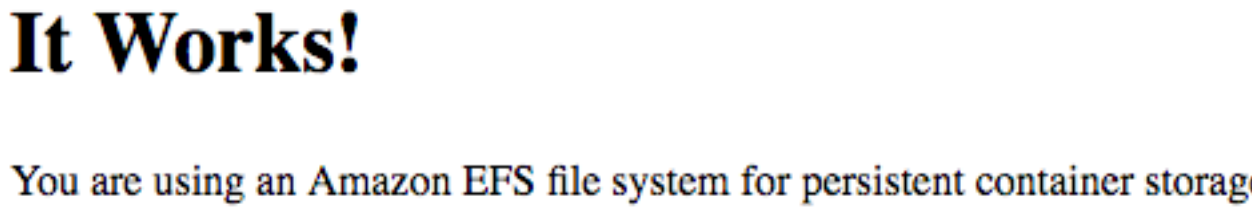
```
sudo bash -c "cat >/efs/html/index.html" <<'EOF'
<html>
    <body>
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container storage.</
p>
    </body>
</html>
EOF
```

Step 7: Run a Task and View the Results

Now that your Amazon EFS file system is available on your container instances and there is web content for the NGINX containers to serve, you can run a task using the task definition that you created earlier. The NGINX web servers serve your simple HTML page. If you update the content in your Amazon EFS file system, those changes are propagated to any containers that have also mounted that file system.

To run a task and view the results

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/.
2. Choose the cluster that you have configured to use Amazon EFS.
3. Choose Tasks, Run new task .
4. For Task Definition, choose the `nginx-efs` taskjob definition that you created earlier and choose Run Task. For more information on the other options in the run task workflow, see 正在运行的任务 (p. 251).
5. Below the Tasks tab, choose the task that you just ran.
6. Expand the container name at the bottom of the page, and choose the IP address that is associated with the container. Your browser should open a new tab with the following message:



Note

If you do not see the message, make sure that the security group for your container instances allows inbound network traffic on port 80.

Amazon ECS 问题排查

您可能需要解决与负载均衡器、任务、服务或容器实例相关的问题。本章将帮助您查找来自 Amazon ECS 容器代理、容器实例上的 Docker 守护程序和 Amazon ECS 控制台中的服务事件日志的诊断信息。

主题

- 首次运行向导启动问题故障排除 (p. 579)
- 检查已停止的任务是否存在错误 (p. 579)
- 服务事件消息 (p. 581)
- 指定的 CPU 或内存值无效 (p. 584)
- 无法提取容器映像错误 (p. 585)
- CannotCreateContainerError: API error (500): devmapper (p. 586)
- 服务负载均衡器问题排查 (p. 587)
- 启用 Docker 调试输出 (p. 588)
- Amazon ECS 日志文件位置 (p. 589)
- Amazon ECS 日志收集器 (p. 590)
- 代理自检诊断 (p. 591)
- Docker 诊断 (p. 593)
- API failures 错误消息 (p. 595)
- 任务 IAM 角色问题排查 (p. 596)

首次运行向导启动问题故障排除

以下错误可能导致 Amazon ECS 首次运行向导无法创建集群。

VpcLimitExceeded

您在尝试完成 Amazon ECS 首次运行向导时可能会收到 `VpcLimitExceeded` 错误。如果是这样的话，您已达到可在区域中创建的 VPC 数量的限制。当您创建 AWS 账户时，对于您可以在每个区域中运行的 VPC 数量有默认限制。有关更多信息，请参阅Amazon VPC 限制。

要解决此问题，您具有以下选项：

- 在每个区域的基础上，请求增加 VPC 服务限制。有关更多信息，请参阅Amazon VPC 限制。
- 删除您的账户中任何未使用的 VPC。有关更多信息，请参阅使用 VPC 和子网。

Important

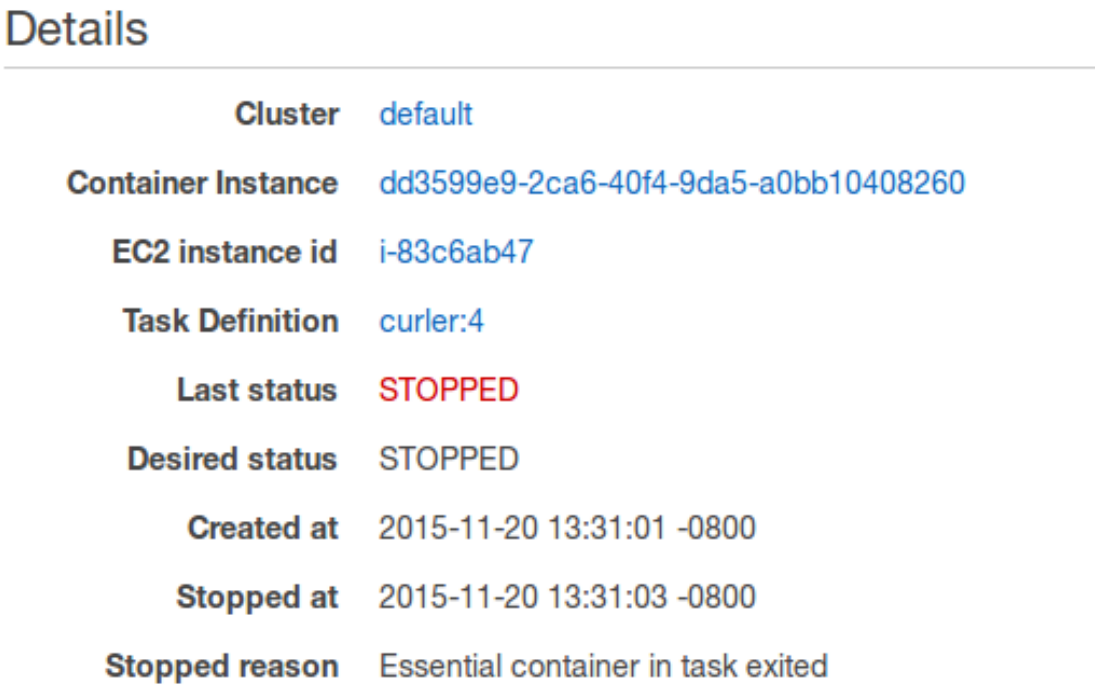
在再次运行首次运行向导之前，可以删除在收到此错误之前已经在此向导中成功创建的任何 Amazon ECS 资源。

检查已停止的任务是否存在错误

如果您在启动任务时遇到问题，则您的任务可能会因错误而停止。例如，在您运行任务时，该任务显示 `PENDING` 状态，然后消失。您可以在 Amazon ECS 控制台中查看诸如此类的错误，方法是显示已停止的任务并检查该任务是否有错误消息。

检查已停止的任务是否存在错误

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters (集群) 页面上，选择已停止的任务所在的集群。
3. 在 Cluster : ***clustername*** (集群: <集群名称>) 页面上，选择 Tasks (任务)。
4. 在 Desired task status (预期任务状态) 表标题中，选择 Stopped (已停止)，然后选择要检查的已停止任务。首先列出最近停止的任务。
5. 在详细信息部分中，检查停止原因字段以查看导致任务停止的原因。



下面列出了一些可能的原因及它们的解释：

Task failed ELB health checks in (elb elb-name)

当前任务未通过针对与任务的服务关联的负载均衡器的 Elastic Load Balancing 运行状况检查。有关更多信息，请参阅服务负载均衡器问题排查 (p. 587)。

Scaling activity initiated by (deployment deployment-id)

当您减少稳定服务的预期数量时，必须停止一些任务才能达到预期数量。通过缩小服务规模停止的任务具有此停止原因。

Host EC2 (instance *id*) stopped/terminated

如果您停止或终止一个具有正在运行的任务的容器实例，这些任务将获得此停止原因。

Container instance deregistration forced by user

如果您强制取消注册一个具有正在运行的任务的容器实例，这些任务将获得此停止原因。

Essential container in task exited

如果任务定义中标记为 `essential` 的容器退出或发生故障，则将导致任务停止。如果主要容器退出是导致任务停止的原因，则Step 6 (p. 580) 会提供有关导致容器停止的原因的更多诊断信息。

6. 如果您有一个已停止的容器，请展开该容器并检查 Status reason 行，以查看导致任务状态发生更改的原因。

Containers

Name	Container Id	Status
curler	3f871451-c9f1-4d6f-a...	STOPPED (CannotPullContainerError: Error: image tutum/bogus)

Details

Status reason CannotPullContainerError: Error: image tutum/bogus:latest not found
Command["/usr/bin/watch","curl","-v","http://amazon-ecs-2004772631.us-west-2.elb.amazonaws.com/"]

在上一个示例中，找不到容器映像名称。如果您拼错了映像名称，则会发生此情况。

如果此检查提供的信息不足，您可以使用 SSH 连接到容器实例并在本地检查 Docker 容器。有关更多信息，请参阅 检查 Docker 容器 (p. 594)。

服务事件消息

如果您正在解决与服务相关的问题，首先应查看服务事件日志中的诊断信息。

在 Amazon ECS 控制台中查看服务事件消息时，将忽略重复的服务事件消息，直至解决了问题根源或者过了六个小时。如果未解决问题根源，您将在六个小时后收到另一个服务事件消息。

在 Amazon ECS 控制台中查看服务事件日志

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters (集群) 页面上，选择服务所在的集群。
3. 在 Cluster : ***clustername*** (集群: <集群名称>) 页面上，选择要检查的服务。
4. 在服务：***servicename*** 页面上，选择事件。

Tasks | Events

Last updated on April 24, 2015 6:33:30 AM (0m ago)

Filter in this page

Viewing 1-10 Events

Event Id	Event Time	Message
22153606-5c...	2015-04-24 06:32:20 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
d863e60c-d3...	2015-04-24 06:30:47 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
dc59a716-b1...	2015-04-24 06:29:14 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
27c37c68-57...	2015-04-24 06:27:41 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
16a36873-8e...	2015-04-24 06:26:08 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
8cee3c0f-693...	2015-04-24 06:24:35 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
2137e914-18...	2015-04-24 06:23:02 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
4142d52d-62...	2015-04-24 06:21:29 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
d4f45e33-766...	2015-04-24 06:19:56 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.
9ad2546b-12...	2015-04-24 06:18:22 -0700	(service sample-webapp) was unable to place a task because the resources could not be found.

5. 检查 Message 列是否存在错误或其他有用信息。

服务事件消息

以下是您可能会在控制台中看到的服务事件消息的示例：

- (service *service-name*) was unable to place a task because no container instance met all of its requirements. (p. 582)
- (service *service-name*) was unable to place a task because no container instance met all of its requirements.最接近的匹配容器实例 *container-instance-id* 没有足够的 CPU 单位可用。 (p. 583)
- (service *service-name*) was unable to place a task because no container instance met all of its requirements.The closest matching container-instance *container-instance-id* encountered error "AGENT". (p. 583)
- (service *service-name*) (instance *instance-id*) is unhealthy in (elb *elb-name*) due to (reason Instance has failed at least the UnhealthyThreshold number of health checks consecutively.) (p. 584)
- (service *service-name*) is unable to consistently start tasks successfully. (p. 584)

(service *service-name*) was unable to place a task because no container instance met all of its requirements.

在上图中，此服务找不到可用的资源，无法添加其他任务。导致出现此情况的可能原因是：

未在您的集群中找到任何容器实例

如果未向您尝试在其中运行任务的集群注册任何容器实例，您将收到此错误。您应向集群添加容器实例。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。

端口不足

如果您的任务使用固定主机端口映射（例如，您的任务对 Web 服务器使用主机上的端口 80），则每个任务必须至少有一个容器实例，因为一次仅一个容器可以使用一个主机端口。您应向集群添加容器实例，或减少所需的任务数。

内存不足

如果您的任务定义指定 1000 MiB 的内存，并且集群中的每个容器实例均具有 1024 MiB 的内存，则只能为每个容器实例运行此任务的一个副本。您可以在任务定义中尝试更少的内存，以便为每个容器实例启动多个任务，或者在集群中启动更多容器实例。

> **Note**
>
> 如果您尝试通过为任务提供尽可能多的用于特定实例类型的内存来最大程度地利用资源，请参阅容器实例内存管理 (p. 190)。

CPU 不足

一个容器实例中，每个 CPU 核心有 1,024 个 CPU 单元。如果您的任务定义指定 1,000 个 CPU 单元，并且集群中的每个容器实例均具有 1,024 个 CPU 单元，则只能为每个容器实例运行此任务的一个副本。您可以在任务定义中尝试更少的 CPU 单元，以便为每个容器实例启动多个任务，或者在集群中启动更多容器实例。

没有足够的可用 ENI 附加点

每个使用 `awsvpc` 网络模式的任务都会收到自己的弹性网络接口 (ENI)，该接口将附加到托管它的容器实例。Amazon EC2 实例对可附加到它们的 ENI 的数量有限制，并且集群中没有带可用 ENI 容量的容器实例。

单个容器实例的 ENI 限制取决于以下条件：

- 如果您尚未选择使用 `awsvpcTrunking` 账户设置，则每个容器实例的 ENI 限制取决于实例类型。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的每个实例类型的每个网络接口的 IP 地址。
- 如果您已选择使用 `awsvpcTrunking` 账户设置，但您在选择使用后未启动使用支持的实例类型的新容器实例，则每个容器实例的 ENI 限制仍将为默认值。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的每个实例类型的每个网络接口的 IP 地址。
- 如果您已选择使用 `awsvpcTrunking` 账户设置，并且您在选择使用后已启动使用支持的实例类型的新容器实例，则其他 ENI 将可用。有关更多信息，请参阅 支持的 Amazon EC2 实例类型 (p. 178)。

有关选择使用 `awsvpcTrunking` 账户设置的更多信息，请参阅 弹性网络接口中继 (p. 175)。

您可以将容器实例添加到您的集群以提供更多可用的网络适配器。

容器实例缺少必需属性

一些任务定义参数需要在容器实例上安装特定的 Docker 远程 API 版本。其他任务定义（例如，日志记录驱动程序选项）需要容器实例使用 `ECS_AVAILABLE_LOGGING_DRIVERS` 代理配置变量注册这些日志记录驱动程序。如果您的任务定义包含一个需要特定容器实例属性的参数，并且您没有可满足此要求的任何可用容器实例，则无法放置任务。有关特定任务定义参数和代理配置变量所需的属性的更多信息，请参阅任务定义参数 (p. 52)和 Amazon ECS 容器代理配置 (p. 217)。

> **Note**
>
> 具有 1.17.0 以前的 Amazon ECS 容器代理版本的 Windows 容器实例默认不支持 `awslogs` 日志驱动程序。如果您无法对 Windows 容器实例使用 `awslogs` 日志驱动程序，请确保您使用的是最新 Amazon ECS-optimized Windows AMI。

(service *service-name*) was unable to place a task because no container instance met all of its requirements.最接近的匹配容器实例 *container-instance-id* 没有足够的 CPU 单位可用。

用于放置任务的最接近的匹配容器实例不会容纳足够的 CPU 单位来满足任务定义中的要求。查看任务定义的任务大小和容器定义参数中的 CPU 要求。

(service *service-name*) was unable to place a task because no container instance met all of its requirements.The closest matching container-instance *container-instance-id* encountered error "AGENT".

用于任务放置的最近匹配容器实例上的 Amazon ECS 容器代理已断开连接。如果您可以通过 SSH 连接到容器实例，则可以查看代理日志；有关更多信息，请参阅 Amazon ECS 容器代理日志 (p. 589)。您还应验证代理是否正在该实例上运行。如果您使用的是Amazon ECS-optimized AMI，则可以尝试通过以下命令停止并重新启动代理：

- 对于Amazon ECS-optimized Amazon Linux 2 AMI：

```
sudo systemctl restart ecs
```

- 对于Amazon ECS-optimized Amazon Linux AMI：

```
sudo stop ecs && sudo start ecs
```

(service *service-name*) (instance *instance-id*) is unhealthy in (elb *elb-name*) due to (reason Instance has failed at least the UnhealthyThreshold number of health checks consecutively.)

此服务已注册到一个负载均衡器，并且未通过负载均衡器运行状况检查。有关更多信息，请参阅服务负载均衡器问题排查 (p. 587)。

(service *service-name*) is unable to consistently start tasks successfully.

此服务包含在连续多次尝试之后仍无法启动的任务。此时，服务计划程序开始逐渐增加重试间隔的时间。您应该排查任务无法启动的原因。有关更多信息，请参阅 服务限制逻辑 (p. 329)。

服务更新后，例如，通过更新的任务定义更新后，服务计划程序恢复正常行为。

指定的 CPU 或内存值无效

在注册任务时，如果指定了无效的 `cpu` 或 `memory` 值，那么您将收到以下错误：

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task. For more information, see the Troubleshooting section of
 the Amazon ECS Developer Guide.
```

要解决此问题，您必须在任务定义中为任务 CPU 和内存指定受支持的值。

虽然在任务定义中可以用 CPU 单元或 vCPU 来表示 `cpu` 值，但在注册任务定义时，该值将转换为一个指示 CPU 单元的整数。如果使用的是 EC2 启动类型，则受支持的值介于 `128` 个 CPU 单元（`0.125` 个 vCPU）和 `10240` 个 CPU 单元（`10` 个 vCPU）之间。如果使用 Fargate 启动类型，则必须使用下表中的值之一，该表决定了 `memory` 参数的受支持值的范围。

虽然在任务定义中可以用 MiB 或 GB 来表示 `memory` 值，但在注册任务定义时，该值将转换为一个指示 MiB 的整数。如果您使用的是 EC2 启动类型，则必须指定一个整数。如果使用 Fargate 启动类型，则必须使用下表中的值之一，该表决定了 `cpu` 参数的受支持值的范围。

Fargate 任务受支持的任务 CPU 和内存值如下：

CPU value	Memory value (MiB)
256 (.25 vCPU)	512 (0.5GB), 1024 (1GB), 2048 (2GB)
512 (.5 vCPU)	1024 (1GB), 2048 (2GB), 3072 (3GB), 4096 (4GB)
1024 (1 vCPU)	2048 (2GB), 3072 (3GB), 4096 (4GB), 5120 (5GB), 6144 (6GB), 7168 (7GB), 8192 (8GB)
2048 (2 vCPU)	Between 4096 (4GB) and 16384 (16GB) in increments of 1024 (1GB)
4096 (4 vCPU)	Between 8192 (8GB) and 30720 (30GB) in increments of 1024 (1GB)

无法提取容器映像错误

以下 Docker 错误表明在创建任务时无法检索指定的容器映像。

连接超时

启动 Fargate 任务时，其弹性网络接口需要到 Internet 的路由来提取容器映像。如果您在启动任务时收到一个类似如下的错误，则原因是到 Internet 的路由不存在：

```
CannotPullContainerError: API error (500): Get https://111122223333.dkr.ecr.us-
east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection"
```

要解决此问题，您可以：

- 对于在公有共子网中的任务，在启动任务时为自动分配公有 IP 指定启用。有关更多信息，请参阅正在运行的任务 (p. 251)。
- 对于在私有子网中的任务，在启动任务时为自动分配公有 IP 指定禁用，然后在 VPC 中配置 NAT 网关，将请求路由到 Internet。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关。有关创建具有公有子网和私有子网的 VPC 的更多信息，包括用于私有子网的 NAT 网关，请参阅Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters (p. 555)。

未找到映像

在容器定义中指定一个 Amazon ECR 映像时，必须使用 ECR 存储库的完整 ARN 或 URI 连同该映像在该存储库中的名称。如果无法找到存储库或映像，您将收到以下错误：

```
CannotPullContainerError: API error (404): repository 111122223333.dkr.ecr.us-
east-1.amazonaws.com/<repo>/<image> not found
```

要解决此问题，请验证存储库的 ARN 或 URI 和映像名称。另外，确保您已经使用任务执行 IAM 角色设置了适当的访问权限。有关任务执行角色的更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。

磁盘空间不足

如果在拉取容器映像时容器实例的根卷没有足够的磁盘空间，您将看到类似以下内容的错误：

```
CannotPullContainerError: write /var/lib/docker/tmp/GetImageBlob111111111: no space
 left on device
```

要解决此问题，请释放磁盘空间。

如果您使用的是Amazon ECS-optimized AMI，则您可以使用以下命令检索文件系统上 20 个最大的文件：

```
du -Sh / | sort -rh | head -20
```

示例输出：

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
...
```

在某些情况下，如上面的示例所示，可以由正在运行的容器填充根卷。如果容器使用的是没有 `max-size` 限制的默认 `json-file` 日志驱动程序，则可能是日志文件占用了大部分已用空间。您可以使用 `docker ps` 命令通过将上面输出中的目录名称映射到容器 ID 来验证哪个容器占用了空间。例如：

```
CONTAINER ID   IMAGE                              COMMAND             CREATED
 STATUS              PORTS                              NAMES
50501b5f4cbf   amazon/amazon-ecs-agent:latest   "/agent"            4 days ago
 Up 4 days                                                ecs-agent
```

默认情况下，使用 `json-file` 日志驱动程序时，Docker 会捕获所有容器的标准输出（和标准错误），并使用 JSON 格式将它们写入到文件中。您可以将 `max-size` 设置为日志驱动程序选项，以防日志文件占用过多空间。有关更多信息，请参阅 Docker 文档中的配置日志记录驱动程序。

下面的容器定义代码段说明此选项的用法：

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "256m"
    }
}
```

如果容器日志占用过多磁盘空间，还有一个替代解决方案是使用 `awslogs` 日志驱动程序。`awslogs` 日志驱动程序将日志发送到 CloudWatch，这将释放容器实例上原本用于容器日志的磁盘空间。有关更多信息，请参阅 使用 awslogs 日志驱动程序 (p. 97)。

CannotCreateContainerError: API error (500): devmapper

下面的 Docker 错误表示容器实例上的精简池存储已满，并且 Docker 守护程序无法创建新容器：

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data blocks
 which is less than minimum required 4454 free data blocks. Create more free space in thin
 pool or use dm.min_free_space option to change behavior
```

默认情况下，`2015.09.d` 版及更高版本中的Amazon ECS-optimized Amazon Linux AMI 使用 8GiB 卷进行启动，该卷用于在 `/dev/xvda` 挂载并装载为文件系统根的操作系统。另外还有一个在 `/dev/xvdcz` 挂载并供 Docker 用于映像和元数据存储的 22 GiB 卷。如果该存储空间被填满，则 Docker 守护程序将无法创建新的容器。

向您的容器实例添加存储的最简单方式是终止现有实例，然后启动具有更大的数据存储卷的新实例。但如果您无法执行此操作，则可以向 Docker 使用的卷组添加存储，然后执行 AMI 存储配置 (p. 151)中的下述步骤来扩展其逻辑卷。

如果您的容器实例存储是填满速度过快，则可采取以下几个措施来减小影响：

- （Amazon ECS 容器代理 1.8.0 及更高版本）减少已停止或已退出的容器在您的容器实例上的保留时间。`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 代理配置变量可设置从任务停止到 Docker 容器被删除所需等待的时间段 (默认情况下，此值为 3 小时)。这将删除 Docker 容器数据。如果此值设置得过低，您可能无法检查已停止的容器或无法在日志删除前查看它们。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。
- 从您的容器实例中删除未运行的容器和未使用的映像。您可以使用以下示例命令来手动删除已停止的容器和未使用的映像。稍后将无法检查已删除的容器，而且必须先重新拉取已删除的映像，然后再从中启动新容器。

  要删除未运行的容器，请在容器实例上执行下面的命令：

  ```
  docker rm $(docker ps -aq)
  ```

要删除未使用的映像，请在容器实例上执行下面的命令：

```
docker rmi $(docker images -q)
```

- 删除容器内未使用的数据块。您可以在任何正在运行的容器上使用下面的命令运行 fstrim 来丢弃该容器文件系统未使用的任何数据块。

```
sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -IZ
 fstrim /proc/Z/root/"
```

服务负载均衡器问题排查

Amazon ECS 服务可向 Elastic Load Balancing 负载均衡器注册任务。负载均衡器配置错误是导致任务停止的常见原因。如果您的已停止任务是由使用负载均衡器的服务启动的，请考虑以下可能原因。

> **Important**
>
> 如果任务属于配置为使用 传统负载均衡器 的服务的一部分，则此类任务不支持容器运行状况检查。Amazon ECS 服务计划程序将忽略 传统负载均衡器后面的处于 `UNHEALTHY` 状态的任务。

`ecsServiceRole` IAM 角色的 IAM 权限不正确

`ecsServiceRole` 允许 Amazon ECS 服务向 Elastic Load Balancing 负载均衡器注册容器实例。您必须具有为此角色的设置的适当权限。有关更多信息，请参阅Amazon ECS 服务计划程序 IAM 角色 (p. 410)。

容器实例安全组

如果您的容器已映射到容器实例上的端口 80，则您的容器实例安全组必须允许端口 80 上的入站流量才能通过负载均衡器运行状况检查。

未为所有可用区配置 Elastic Load Balancing 负载均衡器

您的负载均衡器应配置为使用一个区域中的所有可用区，或至少使用您的容器实例所在的所有可用区。如果服务使用一个负载均衡器，并在驻留在可用区（尚未为该可用区配置使用负载均衡器）中的容器实例上启动一个任务，则该任务永不会通过运行状况检查并且将被终止。

Elastic Load Balancing 负载均衡器运行状况检查配置错误

负载均衡器运行状况检查参数可能过于严格或指向不存在的资源。如果确定容器实例运行状况不佳，则系统会从负载均衡器中删除容器实例。确保验证是否已为服务负载均衡器正确配置以下参数。

Ping 端口

负载均衡器运行状况检查的 Ping Port 值是该检查要确定其运行状况是否正常的容器实例上的端口。如果此端口配置错误，负载均衡器可能会从自身取消注册容器实例。此端口应配置为使用服务的任务定义中用于运行状况检查的容器的 `hostPort` 值。

Ping 路径

此值通常设置为 `index.html`，但如果您的服务未响应该请求，则无法通过运行状况检查。如果您的容器没有 `index.html` 文件，您可以将此值设置为 `/`，以将容器实例的基本 URL 作为目标。

响应超时

这是您的容器必须响应运行状况检查 ping 的时间量。如果此值小于响应所需的时间量，则无法通过运行状况检查。

运行状况检查间隔

这是运行状况检查 ping 之间的时间量。您的运行状况检查间隔越短，您的容器实例就越快达到 Unhealthy Threshold。

不正常阈值

这是您的容器实例被视为不正常之前，允许未通过运行状况检查的次数。如果您的不正常阈值为 2，并且运行状况检查间隔为 30 秒，则您的任务在被视为不正常之前有 60 秒的时间响应运行状况检查 ping。您可以增大不正常阈值或运行状况检查间隔，以便为您的任务提供更多响应时间。

无法更新服务 *servicename*：任务定义中的负载均衡器容器名称或端口已更改

如果您的服务使用一个负载均衡器，则无法更改创建该负载均衡器时为服务定义的负载均衡器配置。如果您更新服务的任务定义，则创建服务时指定的容器名称和容器端口必须保留在任务定义中。

要更改与服务负载均衡器配置关联的负载均衡器名称、容器名称或容器端口，您必须创建新服务。

启用 Docker 调试输出

如果您遇到与 Docker 容器或映像有关的问题，可以在 Docker 守护程序上启用调试模式。启用调试可提供来自守护程序的更详细输出，并且您可以使用此信息来了解有关容器或映像出现问题的原因的更多信息。

启用 Docker 调试模式对于检索从容器注册表（例如 Amazon ECR）发送的错误消息特别有用，在许多情况下，启用调试模式是查看这些错误消息的唯一方式。

Important

此过程是针对Amazon ECS-optimized Amazon Linux AMI 编写的。对于其他操作系统，请参阅 Docker 文档中的启用调试和使用 systemd 控制和配置 Docker。

在Amazon ECS-optimized Amazon Linux AMI 上启用 Docker 守护程序调试模式

1. 连接到您的容器实例。有关更多信息，请参阅 连接到您的容器实例 (p. 180)。
2. 使用文本编辑器（例如 vi）打开 Docker 选项文件。对于Amazon ECS-optimized Amazon Linux AMI，Docker 选项文件位于 `/etc/sysconfig/docker` 中。
3. 查找 Docker 选项语句并将 `-D` 选项添加到用引号引起的字符串。

   Note

   如果 Docker 选项语句以 # 开头，则您需要删除该字符，以取消语句的注释并启用选项。

   对于Amazon ECS-optimized Amazon Linux AMI，Docker 选项语句称作 `OPTIONS`。例如：

```
# Additional startup options for the Docker daemon, for example:
# OPTIONS="--ip-forward=true --iptables=true"
# By default we limit the number of open files per container
OPTIONS="-D --default-ulimit nofile=1024:4096"
```

4. 保存文件并退出您的文本编辑器。
5. 重新启动 Docker 守护程序。

```
sudo service docker restart
```

   输出:

```
Stopping docker:                                          [  OK  ]
Starting docker: .                                  [  OK  ]
```

6. 重新启动 Amazon ECS 代理。

```
sudo start ecs
```

现在，您的 Docker 日志应显示更详细的输出。例如：

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
 server: \"{\\\"errors\\\":[{\\\"code\\\":\\\"DENIED\\\",\\\"message\\\":\\\"User:
 arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
 ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\"Connection\":[]string{\"keep-alive\"}, \"Content-Type\":
[]string{\"application/json; charset=utf-8\"}, \"Date\":[]string{\"Wed, 30 Dec 2015
 21:48:21 GMT\"}, \"Docker-Distribution-Api-Version\":[]string{\"registry/2.0\"},
 \"Content-Length\":[]string{\"235\"}}"
```

Amazon ECS 日志文件位置

Amazon ECS 将日志存储在容器实例的 `/var/log/ecs` 文件夹中。Amazon ECS 容器代理和控制容器实例上的代理状态（启动/停止）的 `ecs-init` 服务提供日志。您可以通过 SSH 连接到容器实例来查看这些日志文件。有关更多信息，请参阅连接到您的容器实例 (p. 180)。

> **Note**
>
> 如果您不确定如何收集容器实例上的所有日志，您可以使用 Amazon ECS 日志收集器。有关更多信息，请参阅Amazon ECS 日志收集器 (p. 590)。

Amazon ECS 容器代理日志

Amazon ECS 容器代理将日志存储在 Linux 实例上的 `/var/log/ecs/ecs-agent.log.`*`timestamp`* 处和 Windows 实例上的 `C:\ProgramData\Amazon\ECS\log\ecs-agent.log.`*`timestamp`* 处。

> **Note**
>
> 您可以通过设置 `ECS_LOGLEVEL=debug` 并重新启动容器代理来提高容器代理日志的详细程度。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。

```
cat /var/log/ecs/ecs-agent.log.2016-08-15-15
```

输出:

```
2016-08-15T15:54:41Z [INFO] Starting Agent: Amazon ECS Agent - v1.12.0 (895f3c1)
2016-08-15T15:54:41Z [INFO] Loading configuration
2016-08-15T15:54:41Z [WARN] Invalid value for task cleanup duration, will be overridden to
 3h0m0s, parsed value 0, minimum threshold 1m0s
2016-08-15T15:54:41Z [INFO] Checkpointing is enabled. Attempting to load state
2016-08-15T15:54:41Z [INFO] Loading state! module="statemanager"
2016-08-15T15:54:41Z [INFO] Detected Docker versions [1.17 1.18 1.19 1.20 1.21 1.22]
2016-08-15T15:54:41Z [INFO] Registering Instance with ECS
2016-08-15T15:54:41Z [INFO] Registered! module="api client"
```

Amazon ECS `ecs-init` 日志

`ecs-init` 过程将日志存储在 `/var/log/ecs/ecs-init.log` 中。

```
cat /var/log/ecs/ecs-init.log
```

输出:

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
```

```
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

任务凭证审核日志的 IAM 角色

当使用 IAM 角色的凭证提供程序向任务提供凭证时，这些请求将记录在 `/var/log/ecs/audit.log.YYYY-MM-DD-HH` 中。

日志条目格式如下：

- 时间戳
- HTTP 响应代码
- 请求来源的 IP 地址和端口号
- 凭证提供程序的相对 URI
- 发出请求的用户代理
- 请求容器所属的任务 ARN
- `GetCredentials` API 名称和版本号
- 容器实例注册到的 Amazon ECS 集群名称
- 容器实例 ARN

示例日志条目如下所示：

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

输出:

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0
 CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials
 1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

Amazon ECS 日志收集器

如果您不确定如何收集容器实例上的所有日志，您可以使用 Amazon ECS 日志收集器。在 GitHub 上提供了适用于 Linux 和 Windows 的日志收集器。此脚本会收集一般操作系统日志以及 Docker 和 Amazon ECS 容器代理日志，这些日志可以帮助解决 AWS 支持案例。然后，它会压缩和存档收集到的信息到单个文件中，该文件可以轻松共享以用于诊断目的。它还支持在 Amazon Linux 变体（例如Amazon ECS-optimized AMI）上为 Docker 守护程序和 Amazon ECS 容器代理启用调试模式。目前，Amazon ECS 日志收集器支持以下操作系统：

- Amazon Linux
- Red Hat Enterprise Linux 7
- Debian 8
- Ubuntu 14.04
- Windows 2016

> **Note**
>
> 对于 Linux 和 Windows，都可以从 GitHub 获得 Amazon ECS 日志收集器的源代码。我们鼓励您针对要包含的更改提交提取请求。但是，Amazon Web Services 当前不支持运行此软件的修改后副本。

下载并运行适用于 Linux 的 Amazon ECS 日志收集器

1. 连接到您的容器实例。有关更多信息，请参阅 连接到您的容器实例 (p. 180)。
2. 下载 Amazon ECS 日志收集器脚本。

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. 运行脚本以收集日志并创建存档。

   Note

   要为 Docker 守护程序和 Amazon ECS 容器代理启用调试模式，可将 `--mode=debug` 选项添加到以下命令。这可能会重新启动 Docker 守护程序，从而终止正在该实例上运行的所有容器。启用调试模式前，应考虑耗尽容器实例并将所有重要任务迁移到其他容器实例。有关更多信息，请参阅 容器实例耗尽 (p. 190)。

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

运行脚本后，您可以在脚本创建的 `collect` 文件夹中检查收集的日志。`collect.tgz` 文件是所有日志的压缩存档，您可以与 AWS 支持共享该文件以帮助诊断。

下载并运行适用于 Windows 的 Amazon ECS 日志收集器

1. 连接到您的容器实例。有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Windows 实例）中的连接到您的 Windows 实例。
2. 使用 PowerShell 下载 Amazon ECS 日志收集器脚本。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-logs-collector.ps1
```

3. 运行脚本以收集日志并创建存档。

   Note

   要为 Docker 守护程序和 Amazon ECS 容器代理启用调试模式，可将 `-RunMode debug` 选项添加到以下命令。这会重新启动 Docker 守护程序，从而终止正在该实例上运行的所有容器。启用调试模式前，应考虑耗尽容器实例并将所有重要任务迁移到其他容器实例。有关更多信息，请参阅 容器实例耗尽 (p. 190)。

```
.\ecs-logs-collector.ps1
```

运行脚本后，您可以在脚本创建的 `collect` 文件夹中检查收集的日志。`collect.tgz` 文件是所有日志的压缩存档，您可以与 AWS 支持共享该文件以帮助诊断。

代理自检诊断

Amazon ECS 代理自检 API 可提供有用的诊断信息。例如，您可以使用代理自检 API 获取任务中的容器的 Docker ID。您可以使用 SSH 连接到容器实例来使用代理自检 API。有关更多信息，请参阅连接到您的容器实例 (p. 180)。

Important

容器实例必须具有能够访问 Amazon ECS 的 IAM 角色才能访问自检 API。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

下面的示例介绍两个任务，一个任务目前正在运行，另一个任务已停止。

Note

以下命令通过 python -mjson.tool 传输以提高可读性。

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

输出:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1095  100  1095    0     0   117k      0 --:--:-- --:--:-- --:--:--  133k
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-
a8d14064fd24",
            "Containers": [
                {
                    "DockerId":
 "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
                    "DockerName": "ecs-console-sample-app-static-6-simple-
app-86caf9bcabe3e9c61600",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
                    "DockerName": "ecs-console-sample-app-static-6-busybox-
ce83ce978a87a890ab01",
                    "Name": "busybox"
                }
            ],
            "Family": "console-sample-app-static",
            "KnownStatus": "STOPPED",
            "Version": "6"
        },
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
            "Containers": [
                {
                    "DockerId":
 "dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
                    "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
                    "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
                    "Name": "busybox"
                }
            ],
            "DesiredStatus": "RUNNING",
            "Family": "console-sample-app-static",
            "KnownStatus": "RUNNING",
            "Version": "6"
        }
    ]
```

```
}
```

在上面的示例中，已停止的任务 (*090eff9b-1ce3-4db6-848a-a8d14064fd24*) 具有两个容器。您可以使用 docker inspect ***container-ID*** 查看每个容器的详细信息。有关更多信息，请参阅Amazon ECS 容器代理自检 (p. 244)。

Docker 诊断

Docker 提供了几种诊断工具，可帮助您解决与容器和任务相关的问题。有关所有可用的 Docker 命令行实用工具的更多信息，请参阅 Docker 文档中的 Docker 命令行主题。您可以通过使用 SSH 连接到容器实例来访问 Docker 命令行实用工具。有关更多信息，请参阅 连接到您的容器实例 (p. 180)。

Docker 容器报告的退出代码也可提供一些诊断信息（例如，退出代码 137 表示容器已收到 `SIGKILL` 信号）。有关更多信息，请参阅 Docker 文档中的退出状态。

列出 Docker 容器

您可以在容器实例上使用 docker ps 命令列出正在运行的容器。在以下示例中，仅 Amazon ECS 容器代理正在运行。有关更多信息，请参阅 Docker 文档中的 docker ps。

```
docker ps
```

输出:

```
CONTAINER ID        IMAGE                               COMMAND             CREATED
  STATUS              PORTS                           NAMES
cee0d6986de0        amazon/amazon-ecs-agent:latest    "/agent"            22 hours ago
  Up 22 hours         127.0.0.1:51678->51678/tcp    ecs-agent
```

您可以使用 docker ps -a 命令来查看所有容器（甚至是已停止或已终止的容器）。这对于列出意外停止的容器很有用。在以下示例中，容器 `f7f1f8a7a245` 已在 9 秒前退出，因此不会显示在未带 `-a` 标记的 docker ps 输出中。

```
docker ps -a
```

输出:

```
CONTAINER ID        IMAGE                                          COMMAND
 CREATED             STATUS                          PORTS                          NAMES
db4d48e411b1        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"       19
 seconds ago                                                                      ecs-console-
sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700
f7f1f8a7a245        busybox:buildroot-2014.02                   "\"sh -c '/bin/sh -c   22
 hours ago        Exited (137) 9 seconds ago                                      ecs-console-
sample-app-static-6-busybox-ce83ce978a87a890ab01
189a8ff4b5f0        httpd:2                                     "httpd-foreground"     22
 hours ago        Exited (137) 40 seconds ago                                     ecs-console-
sample-app-static-6-simple-app-86caf9bcabe3e9c61600
0c7dca9321e3        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"       22
 hours ago                                                                        ecs-console-
sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0        amazon/amazon-ecs-agent:latest              "/agent"               22
 hours ago        Up 22 hours                   127.0.0.1:51678->51678/tcp   ecs-agent
```

查看 Docker 日志

您可以使用 docker logs 命令查看容器的 STDOUT 和 STDERR 流。在此示例中，为简洁起见，将为 *dc7240fe892a* 容器显示日志，并通过 head 命令传输日志。有关更多信息，请转至 Docker 文档中的 docker logs。

> **Note**
>
> 如果使用默认 json 日志驱动程序，则 Docker 日志仅在容器实例上可用。如果已配置任务使用 awslogs 日志驱动程序，则容器日志在 CloudWatch Logs 中可用。有关更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。

```
docker logs dc7240fe892a | head
```

输出:

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472] AH00489:
 Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
 Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

检查 Docker 容器

如果拥有容器的 Docker ID，可以使用 docker inspect 命令检查该容器。通过检查容器，可提供在其中启动容器的环境的最详细视图。有关更多信息，请参阅 Docker 文档中的 docker inspect。

```
docker inspect dc7240fe892a
```

输出:

```
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "httpd-foreground"
        ],
        "CpuShares": 10,
        "Cpuset": "",
        "Domainname": "",
```

```
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/
apache2/bin",
            "HTTPD_PREFIX=/usr/local/apache2",
            "HTTPD_VERSION=2.4.12",
            "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
        ],
        "ExposedPorts": {
            "80/tcp": {}
        },
        "Hostname": "dc7240fe892a",
...
```

API failures 错误消息

有时，通过 Amazon ECS 控制台或 AWS CLI 触发的 API 调用退出时出现 failures 错误消息。下面说明了针对每个 API 调用的可能的 API failures 错误消息。故障在特定资源上发生，而圆括号中的资源是与故障关联的资源。

许多资源都特定于区域，因此，在设置控制台时，确保区域和资源对应。另外，确保使用 --region *region* 选项将 AWS CLI 命令发送到正确区域。

- DescribeClusters

  MISSING（集群 ID）

  找不到您的集群。集群名称的拼写可能不正确，或者指定了错误的区域。
- DescribeInstances

  MISSING（容器实例 ID）

  您尝试描述的容器实例不存在。可能指定了错误的集群或区域，或者容器实例 ARN 或 ID 拼写错误。
- DescribeServices

  MISSING（服务 ID）

  您尝试描述的服务不存在。可能指定了错误的集群或区域，或者容器实例 ARN 或 ID 拼写错误。
- DescribeTasks

  MISSING（任务 ID）

  您尝试描述的任务不存在。可能指定了错误的集群或区域，或者任务 ARN 或 ID 拼写错误。
- RunTask 或者 StartTask

  RESOURCE:*（容器实例 ID）

  任务请求的资源在给定容器实例上不可用。如果资源是 CPU、内存、端口或弹性网络接口，则您可能需要将容器实例添加到集群。对于 RESOURCE:ENI 错误，说明集群没有任何可用的弹性网络接口附加点，而这是使用 awsvpc 网络模式的任务所必需的。Amazon EC2 实例对可以附加到它们的网络接口数量有限制，并且主网络接口算作一个。有关每个实例类型支持的网络接口数量的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的每个实例类型的每个网络接口的 IP 地址。

  RESOURCE:GPU

  任务请求的 GPU 数量在给定容器实例上不可用。您可能需要将支持 GPU 的容器实例添加到集群。有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。

  AGENT（容器实例 ID）

  您尝试在其上启动任务的容器实例有一个目前已断开连接的代理。为防止任务放置的等待时间延长，已拒绝该请求。

`ATTRIBUTE`（容器实例 ID）

您的任务定义包含一个需要容器实例上未提供的特定容器实例属性的参数。例如，如果您的任务使用 `awsvpc` 网络模式，但您指定的子网中没有包含 `ecs.capability.task-eni` 属性的实例。有关特定任务定义参数和代理配置变量需要哪些属性的更多信息，请参阅任务定义参数 (p. 52)和 Amazon ECS 容器代理配置 (p. 217)。

> **Note**
>
> 具有 1.17.0 以前的 Amazon ECS 容器代理版本的 Windows 容器实例默认不支持 `awslogs` 日志驱动程序。如果您无法对 Windows 容器实例使用 `awslogs` 日志驱动程序，请确保您使用的是最新 Amazon ECS-optimized Windows AMI。

- `StartTask`

  `MISSING`（容器实例 ID）

  您尝试在其上启动任务的容器实例不存在。可能指定了错误的集群或区域，或者容器实例 ARN 或 ID 拼写错误。

  `INACTIVE`（容器实例 ID）

  您尝试在其上启动任务的容器实例之前已从 Amazon ECS 取消注册且无法使用。

任务 IAM 角色问题排查

如果在为自己集群中的任务配置 IAM 角色时遇到问题，您可以尝试使用这个已知良好的配置来帮助调试自己的配置。

下面的过程可帮助您：

- 创建 CloudWatch Logs 日志组来存储测试日志。
- 创建具有完全 Amazon ECS 权限的任务 IAM 角色。
- 使用与任务 IAM 角色兼容的已知良好的 AWS CLI 配置注册任务定义。
- 从该任务定义运行任务，以测试您的容器实例是否支持任务的 IAM 角色。
- 在 CloudWatch Logs 中查看该任务的容器日志，以验证它是否正常工作。

用已知良好的配置测试任务的 IAM 角色

1. 创建一个称作 `ecs-tasks` 的 CloudWatch Logs 日志组。
   a. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
   b. 在左侧导航窗格中，依次选择日志、操作和创建日志组。
   c. 对于日志组名称，输入 `ecs-tasks`，然后选择创建日志组。
2. 为任务创建一个要使用的 IAM 角色。
   a. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
   b. 在导航窗格中，选择 Roles 和 Create role。
   c. 对于 Select type of trusted entity (选择可信任的实体的类型)，请选择 Elastic Container Service。
   d. 对于 Select your use case (选择您的使用案例)，请选择 Elastic Container Service Task (Elastic Container Service 任务)、Next: Permissions (下一步: 权限)。
   e. 在附加的权限策略页面上，选择 AmazonEC2ContainerServiceFullAccess，然后选择下一步：审核。
   f. 在 Review 页面上，对于 Role name，输入 `ecs-task-full-access`，然后选择 Create role。

3. 注册使用您的新角色的任务定义。

   a. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
   b. 在导航窗格中，选择 Task Definitions。
   c. 在 Task Definitions (任务定义) 页面上，选择 Create new Task Definition (创建新任务定义)。
   d. 在 Select launch type compatibility 页面上，选择 EC2、Next step。
   e. 滚动到页面底部，选择 Configure via JSON (通过 JSON 配置)。
   f. 将下面的示例任务定义 JSON 粘贴到文本区域 (替换预填充的 JSON)，然后选择 Save。

      Note

      将 `awslogs-region` 值替换成您在其中创建了 CloudWatch Logs 日志组的区域。

```
{
    "taskRoleArn": "ECS-task-full-access",
    "containerDefinitions": [
        {
            "memory": 128,
            "essential": true,
            "name": "amazonlinux",
            "image": "amazonlinux",
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": [
                "yum install -y aws-cli; aws ecs list-tasks --region us-west-2"
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "ecs-tasks",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "iam-role-test"
                }
            }
        }
    ],
    "family": "iam-role-test",
    "requiresCompatibilities": [
        "EC2"
    ],
    "volumes": [],
    "placementConstraints": [],
    "networkMode": null,
    "memory": null,
    "cpu": null
}
```

   g. 验证您的信息并选择 Create。
4. 从您的任务定义运行任务。

   a. 在 Task Definition: iam-role-test 注册确认页面上，依次选择 Actions、Run Task。
   b. 在 Run Task 页面上，选择 EC2 启动类型，选择一个集群，然后选择 Run Task 以运行您的任务。
5. 在 CloudWatch Logs 控制台中查看容器日志。

   a. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
   b. 在左侧导航窗格中，选择 Logs。
   c. 选择 `ecs-tasks` 日志组。
   d. 选择最新的日志流。

e. 向下滚动以查看日志流的最后几行。您应该看到 aws ecs list-tasks 命令的输出。

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/d48feb62-46e2-4cbc-a36b-
e0400b993d1d"
    ]
}
```

如果您收到一条“`Unable to locate credentials`”错误，则以下是可能的原因。

- 任务的 IAM 角色功能未在您的容器实例上启用。有关更多信息，请参阅在容器实例上启用任务 IAM 角色 (p. 420)。
- 凭证 URL 已受限。您可以使用 `ECS_TASK_METADATA_RPS_LIMIT` 容器代理参数来配置限制。有关更多信息，请参阅 Amazon ECS 容器代理配置 (p. 217)。

Windows 容器

Amazon ECS 现在支持使用Amazon ECS-optimized Windows AMI 启动的容器实例上的 Windows 容器。

Windows 容器实例使用其自己的 Amazon ECS 容器代理版本。在Amazon ECS-optimized Windows AMI 上，Amazon ECS 容器代理在主机上作为一项服务运行。与 Linux 平台不同，此代理不在容器内部运行，因为它使用主机的注册表和指定管道 (位于 `\\.\pipe\docker_engine` 中) 与 Docker 守护程序进行通信。

GitHub 上提供了 Amazon ECS 容器代理的源代码。我们鼓励您针对要包含的更改提交提取请求。但是，我们当前不支持运行此软件的修改后副本。您可以在我们的 GitHub 问题页面上查看有关 Amazon ECS 和 Windows 的待解决问题。

Amazon ECS 提供了已针对 Windows 容器进行了优化的 AMI，包括以下变体。

- Amazon ECS-optimized Windows 2019 AMI – 推荐用于在 Windows 操作系统上启动您的 Amazon ECS 容器实例。
- Amazon ECS-optimized Windows 2016 AMI – 可用于在 Windows 操作系统上启动您的 Amazon ECS 容器实例。

主题

- Windows 容器注意事项 (p. 599)
- Windows 容器入门 (p. 600)
- Windows 任务定义 (p. 605)
- 任务的 Windows IAM 角色 (p. 608)
- 将 Windows 映像推送到 Amazon ECR (p. 608)

Windows 容器注意事项

下面是您应了解的有关 Windows 容器和 Amazon ECS 的一些事项。

- Windows 容器无法在 Linux 容器实例上运行，反之亦然。要确保正确放置 Windows 和 Linux 任务，您应将 Windows 容器实例和 Linux 容器实例放置在单独的集群中，并且仅将 Windows 任务放置在 Windows 集群上。您可以设置以下放置约束，确保 Windows 任务定义仅放置在 Windows 实例上：`memberOf(ecs.os-type=='windows')`。
- 仅支持使用 EC2 启动类型的任务使用 Windows 容器。对于 Windows 容器，Fargate 启动类型目前不受支持。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型 (p. 81)。
- Windows 容器和容器实例并不完全支持适用于 Linux 容器和容器实例的所有任务定义和参数。对于某些参数，它们完全不受支持，而其他参数在 Windows 和 Linux 上的行为不相同。有关更多信息，请参阅 Windows 任务定义 (p. 605)。
- 任务的 IAM 角色功能需要您将 Windows 容器实例配置为在启动时允许该功能，并且您的容器在使用此功能时必须运行某些提供的 PowerShell 代码。有关更多信息，请参阅 任务的 Windows IAM 角色 (p. 608)。
- 任务的 IAM 角色功能使用凭证代理来向容器提供凭证。此凭证代理占用了容器实例上的端口 80，因此，如果您使用 IAM 角色处理任务，则端口 80 不可用于任务。对于 Web 服务容器，您可以使用 应用程序负载均衡器 和动态端口映射来向容器提供标准 HTTP 端口 80 连接。有关更多信息，请参阅 服务负载均衡 (p. 284)。

- 由于 Windows 服务器 Docker 映像很大 (9 GiB)，因此您的容器实例所需的存储空间大于 Linux 容器实例所需的存储空间，后者通常具有较小的映像。

Windows 容器入门

本教程演示如何使用Amazon ECS-optimized Windows AMI 手动在 Amazon ECS 上运行 Windows 容器。为 Windows 容器实例创建一个集群，在您的集群中启动一个或多个容器实例，注册一个使用 Windows 容器映像的任务定义，创建一项使用该任务定义的服务，然后查看容器运行的示例网页。

主题

- 步骤 1：创建 Windows 集群 (p. 600)
- 步骤 2：在集群中启动 Windows 容器实例 (p. 600)
- 步骤 3：注册 Windows 任务定义 (p. 603)
- 步骤 4：使用您的任务定义创建服务 (p. 604)
- 步骤 5：查看您的服务 (p. 604)

步骤 1：创建 Windows 集群

您应为 Windows 容器创建新集群。Linux 容器实例无法运行 Windows 容器，反之亦然，因此最好是通过在单独的集群中运行 Windows 容器实例和 Linux 容器实例来正确地放置任务。在本教程中，您为 Windows 容器创建一个名为 `windows` 的集群。

使用 AWS 管理控制台创建集群

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Clusters。
3. 在 Clusters 页面上，选择 Create Cluster。
4. 选择 EC2 Windows + Networking，然后选择 Next step。
5. 在 Cluster name (集群名称) 中，为集群输入名称（在本示例中，集群的名称是 `windows`）。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。
6. 依次选择 Create an empty cluster、Create。

使用 AWS CLI创建集群

- 您可以使用下面的命令通过 AWS CLI 创建集群：

```
aws ecs create-cluster --cluster-name windows
```

步骤 2：在集群中启动 Windows 容器实例

您可以使用 AWS 管理控制台启动 Windows 容器实例，如本主题中所述。开始之前，请确保您已完成Amazon ECS 的设置 (p. 7)中的步骤。启动实例之后，您可以使用实例来运行任务。

启动 Windows 容器实例

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中，选择要使用的区域。
3. 从控制台控制面板中，选择 Launch Instance。

4. 在 Choose an Amazon Machine Image (AMI) (选择 Amazon 系统映像 (AMI)) 页面上的 Search community AMIs (搜索社区 AMI) 字段中键入 ECS_Optimized，然后按 Enter 键。选择 Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19 AMI 旁边的 Select (选择)。

   下表按区域列出了当前的 Amazon ECS-optimized Windows 2019 AMI ID。

Region	AMI ID
us-east-2	ami-05c650b3e1759298f
us-east-1	ami-00f3f16f1d05e53a9
us-west-1	ami-08f56c127cc9ae9c4
us-west-2	ami-0c37f25e59d752c61
ap-northeast-1	ami-0771bfb73bc362295
ap-northeast-2	ami-0849ba95174008d74
ap-south-1	ami-0dfe1bb9ed2dd2ff1
ap-southeast-1	ami-06daa78e84b800914
ap-southeast-2	ami-071e43d09ba6f2479
ca-central-1	ami-0ed093f8cd0bb6f64
eu-central-1	ami-020bded18f4df2c7d
eu-north-1	ami-0269f4d7fa129ddcb
eu-west-1	ami-0e3604f46f9ccc506
eu-west-2	ami-0746c037f8889f6e8
eu-west-3	ami-022ae3cb06a066371
sa-east-1	ami-094ff1452e5849bff
us-gov-east-1	ami-0e238cac0ef640949
us-gov-west-1	ami-13fcbd72

   Note

   Windows Server 2019 和 Windows Server 2016 都有经过 Amazon ECS 优化的 AMI。有关更多信息，请参阅如何启动最新的 Amazon ECS-Optimized Amazon Linux AMI (p. 137)。

5. 在 Choose an Instance Type (选择实例类型) 页面上，您可以选择实例的硬件配置。默认情况下会选择 `t2.micro` 实例类型。您选择的实例类型决定了可用于运行您的任务的资源。
6. 选择 Next: Configure Instance Details。
7. 在配置实例详细信息页面上，根据是否允许从公共 Internet 访问实例来设置自动分配公有 IP 复选框。如果实例应可从 Internet 进行访问，请确保自动分配公有 IP 字段设置为启用。如果不应可从 Internet 访问您的实例，请选择 Disable。

   Note

   容器实例需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的容器实例完成。

有关接口 VPC 终端节点的更多信息，请参阅Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink) (p. 337)。
如果您没有配置接口 VPC 终端节点并且您的容器实例没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关和本指南中的 HTTP 代理配置 (p. 245)。有关更多信息，请参阅Tutorial: Creating a VPC with Public and Private Subnets for Your Clusters (p. 555)。

8. 在 Configure Instance Details 页面上，选择您在`ecsInstanceRole`中为您的容器实例创建的 IAM roleAmazon ECS 的设置 (p. 7) 值。

   Important

   如果未使用适当的 IAM 权限启动容器实例，则 Amazon ECS 代理不会连接到集群。有关更多信息，请参阅Amazon ECS 容器实例 IAM 角色 (p. 404)。

9. 展开 Advanced Details 部分，并将提供的用户数据 PowerShell 脚本粘贴到 User data 字段。默认情况下，此脚本将容器实例注册到您之前创建的 `windows` 集群。要在另一个集群而非 `windows` 中启动，可将以下脚本中的红色文本替换为您的集群名称。

   Note

   启用任务的 IAM 角色需要使用 `-EnableTaskIAMRole` 选项。有关更多信息，请参阅任务的 Windows IAM 角色 (p. 608)。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
</powershell>
```

10. 选择 Next: Add Storage。
11. 在 Add Storage (添加存储) 页面上，为您的容器实例配置存储。Windows 操作系统和容器映像很大 (对于 Windows 服务器核心基础层来说，约为 9 GiB)，仅几个映像和容器就会快速达到 Amazon ECS-optimized Windows AMI 的默认 50 GiB 卷大小。更大的根卷大小（例如，200 GiB）允许实例上具有更多的容器和映像。

    您可以选择增大或减小实例的卷大小以满足您的应用程序需求。

12. 选择 Review and Launch。
13. 在 Review Instance Launch 页面上的 Security Groups 下，您将看到向导为您创建并选择了安全组。默认情况下，您应通过端口 3389 来实现 RDP 连接。要让容器接收来自 Internet 的入站流量，也需要开放这些端口。

    a. 选择 Edit security groups。
    b. 在 Configure Security Group 页面上，确保选择 Create a new security group 选项。
    c. 为容器可能需要的任何其他端口添加规则，然后选择 Review and Launch (审核并启动)。本演练稍后的示例任务定义将使用端口 8080，因此您应对 Anywhere (任意位置) 开放此端口。

14. 在 Review Instance Launch 页面上，选择 Launch。
15. 在 Select an existing key pair or create a new key pair 对话框中，选择 Choose an existing key pair，然后选择您在开始设置时创建的密钥对。

    准备好后，选中确认字段，然后选择 Launch Instances。

16. 确认页面会让您知道自己的实例已启动。选择 View Instances 以关闭确认页面并返回控制台。
17. 在 Instances (实例) 屏幕上，您可以查看您实例的状态。启动实例只需很短的时间。启动实例时，其初始状态为 `pending`。实例启动后，其状态变为 `running`，并且会收到一个公有 DNS 名称。（如果 Public DNS 列已隐藏，请选择 Show/Hide 图标，然后选择 Public DNS。）
18. 在启动实例后，您可以在 Amazon ECS 控制台中查看集群，了解容器实例是否已注册到该集群。

    Note

    将 Windows 容器实例注册到集群最多需要花费 15 分钟的时间。

步骤 3：注册 Windows 任务定义

您必须先注册任务定义，然后才能在 Amazon ECS 集群中运行 Windows 容器。以下任务定义示例在具有 `microsoft/iis` 容器映像的容器实例的端口 8080 上显示一个简单网页。

使用 AWS 管理控制台注册示例任务定义

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在导航窗格中，选择 Task Definitions。
3. 在 Task Definitions (任务定义) 页面上，选择 Create new Task Definition (创建新任务定义)。
4. 在 Select launch type compatibilities (选择启动类型兼容性) 页面上，选择 EC2、Next step (下一步)。

   Note

   Fargate 启动类型与 Windows 容器不兼容。
5. 滚动到页面底部，选择 Configure via JSON (通过 JSON 配置)。
6. 将下面的示例任务定义 JSON 粘贴到文本区域 (替换预填充的 JSON)，然后选择 Save。

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "microsoft/iis",
      "cpu": 512,
      "entryPoint":["powershell", "-Command"],
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -ItemType file -
Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>' -Force ; C:\
\ServiceMonitor.exe w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ],
      "memory": 768,
      "essential": true
    }
  ]
}
```

7. 验证您的信息并选择 Create。

使用 AWS CLI注册示例任务定义

1. 创建名为 `windows-simple-iis.json` 的文件。
2. 使用您偏好的文本编辑器打开此文件，并将上面的示例 JSON 添加到此文件，然后保存文件。
3. 使用 AWS CLI 运行以下命令以将任务定义注册到 Amazon ECS。

   Note

   确保将 AWS CLI 配置为使用 Windows 集群所在的区域，或者将 `--region` *`your_cluster_region`* 选项添加到命令。

```
aws ecs register-task-definition --cli-input-json file://windows-simple-iis.json
```

步骤 4：使用您的任务定义创建服务

在注册任务定义后，您可以使用任务定义在集群中放置任务。以下过程使用任务定义创建一个服务并将一个任务放置在集群中。

使用控制台从任务定义创建服务

1. 在 Task Definition: windows-simple-iis 注册确认页面上，依次选择 Actions、Create Service。
2. 在 Create Service (创建服务) 页面上，输入以下信息，然后选择 Create Service (创建服务)。
   - Launch type (启动类型)：`EC2`
   - Cluster (集群)：windows
   - Service name (服务名称)：windows-simple-iis
   - Service type (服务类型)：`REPLICA`
   - Number of tasks (任务数)：1
   - Deployment type (部署类型)：滚动更新

使用 AWS CLI 从任务定义创建服务

- 使用 AWS CLI 运行以下命令来创建服务。

```
aws ecs create-service --cluster windows --task-definition windows-simple-iis --
desired-count 1 --service-name windows-simple-iis
```

步骤 5：查看您的服务

当您的服务在集群中启动任务后，您可以查看服务并在浏览器中打开 IIS 测试页面以验证容器是否正在运行。

> **Note**
>
> 下载容器实例并提取 Windows 容器基础层最多需要花费 15 分钟的时间。

查看您的服务

1. 在 https://console.aws.amazon.com/ecs/ 上打开 Amazon ECS 控制台。
2. 在 Clusters 页面上，选择 windows 集群。
3. 在 Services 选项卡上，选择 windows-simple-iis 服务。
4. 在 Service: windows-simple-iis 页面上，选择服务中的任务的任务 ID。
5. 在 Task 页面上，展开 `iis` 容器以查看其信息。
6. 在容器的 Network bindings 中，您应看到 External Link IP 地址和端口组合链接。选择该链接以在浏览器中打开 IIS 测试页面。

Windows 任务定义

Windows 容器和容器实例并不完全支持适用于 Linux 容器和容器实例的所有任务定义和参数。对于某些参数，它们完全不受支持，而其他参数在 Windows 和 Linux 上的行为不相同。

Windows 任务定义参数

以下列表说明 Windows 容器不支持哪些参数或哪些参数在该容器中的行为与其在 Linux 容器中的行为不同。有关这些参数与 Amazon ECS 相关的更多信息，请参阅任务定义参数 (p. 52)。

`taskRoleArn`

支持：是

Windows 上任务的 IAM 角色需要在您启动Amazon ECS-optimized Windows AMI 时设置 `-EnableTaskIAMRole` 选项。此外，您的容器必须运行一些配置代码才能利用此功能。有关更多信息，请参阅 任务的 Windows IAM 角色 (p. 608)。

`networkMode`

支持：否

Docker for Windows 使用的网络模式不同于 Docker for Linux。在将任务定义注册到 Windows 容器时，您不得指定网络模式。如果使用控制台将任务定义注册到 Windows 容器，您必须选择 `<default>` 网络模式对象。

`containerDefinitions`

支持：是

附加说明：并非所有容器定义参数都受支持。查看以下列表来了解各个参数支持。

`portMappings`

支持：有限

Windows 上的端口映射使用 `NetNAT` 网关地址而非 `localhost`。没有针对 Windows 上的端口映射的回环，因此，您无法从主机自身访问容器的映射端口。

`cpu`

支持：是

Amazon ECS 将采用针对 Linux 容器的处理方式来处理此参数：如果您向容器提供 500 个 CPU 份额，则在放置任务后，将从容器实例上的可用资源中减去这些 CPU 份额。但是，在 Windows 容器

实例上，此 CPU 限制将作为绝对限制或配额强制实施。Windows 容器只能访问任务定义中所述的指定的 CPU 量。

`disableNetworking`

支持：否

`dnsServers`

支持：否

`dnsSearchDomains`

支持：否

`dockerSecurityOptions`

支持：否

`extraHosts`

支持：否

`links`

支持：否

`mountPoints`

支持：有限

Windows 容器可在 `$env:ProgramData` 所在的驱动器上装载整个目录。Windows 容器无法在其他驱动器上装载目录，并且装载点不能跨驱动器。

`linuxParameters`

支持：否

`privileged`

支持：否

`readonlyRootFilesystem`

支持：否

`user`

支持：否

`ulimits`

支持：否

`volumes`

支持：是

`name`

支持：是

`dockerVolumeConfiguration`

支持：否

`host`

支持：有限

Windows 容器可在 `$env:ProgramData` 所在的驱动器上装载整个目录。Windows 容器无法在其他驱动器上装载目录，并且装载点不能跨驱动器。例如，您可以装载 `C:\my\path:C:\my\path` 和 `D:\:D:\`，而不能装载 `D:\my\path:C:\my\path` 或 `D:\:C:\my\path`。

`cpu`

支持：否

Windows 容器将忽略任务级 CPU。我们建议为 Windows 容器指定容器级 CPU。

`memory`

支持：否

Windows 容器将忽略任务级内存。我们建议为 Windows 容器指定容器级内存。

`proxyConfiguration`

支持：否

`ipcMode`

支持：否

`pidMode`

支持：否

Windows 示例任务定义

以下示例任务定义可帮助您在 Amazon ECS 上开始使用 Windows 容器。

Example 适用于 Windows 的 Amazon ECS 控制台示例应用程序

以下任务定义是 Amazon ECS 的首次运行向导中生成的 Amazon ECS 控制台示例应用程序；它已转为使用 `microsoft/iis` Windows 容器映像。

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "microsoft/iis",
      "cpu": 512,
      "entryPoint":["powershell", "-Command"],
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file -Value '<html>
 <head> <title>Amazon ECS Sample App</title> <style>body {margin-top: 40px; background-
color: #333;} </style> </head><body> <div style=color:white;text-align:center> <h1>Amazon
 ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a
 container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ],
      "memory": 1024,
      "essential": true
    }
  ]
}
```

任务的 Windows IAM 角色

使用 Windows 功能的任务的 IAM 角色需要额外配置，但此配置很大程度上类似于在 Linux 容器实例上启用任务的 IAM 角色。必须满足以下要求才能为 Windows 容器启用任务的 IAM 角色。

- 在启动容器实例时，您必须通过在容器实例用户数据脚本中设置 `-EnableTaskIAMRole` 选项来启用此功能。例如：

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
</powershell>
```

- 您必须使用任务的 IAM 角色的容器引导脚本 (p. 608)中提供的联网命令来引导容器。
- 您必须为任务创建 IAM 角色和策略。有关更多信息，请参阅为任务创建 IAM 角色和策略 (p. 420)。
- 您的容器必须使用支持任务的 IAM 角色的 AWS 开发工具包。有关更多信息，请参阅使用支持的 AWS 开发工具包 (p. 422)。
- 您必须指定在注册任务定义时为任务创建的 IAM 角色，或者在运行任务时作为覆盖指定。有关更多信息，请参阅为任务指定 IAM 角色 (p. 422)。
- 任务的 IAM 角色凭证提供程序在容器实例上使用端口 80，因此，如果在容器实例上启用任务的 IAM 角色，则容器无法将端口 80 用于任何端口映射中的主机端口。要在端口 80 上公开容器，建议您为这些容器配置一个使用负载平衡功能的服务。您可以在负载平衡器上使用端口 80，并且流量可传送到容器实例上的另一个主机端口。有关更多信息，请参阅 服务负载均衡 (p. 284)。

任务的 IAM 角色的容器引导脚本

必须先使用所需的联网命令引导容器，然后容器才能访问容器实例上的凭证代理来获得凭证。当容器启动时，应在容器上运行以下代码示例脚本。

```
# Copyright 2014-2016 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

$gateway = (Get-NetRoute | Where { $_.DestinationPrefix -eq '0.0.0.0/0' } | Sort-Object
 RouteMetric | Select NextHop).NextHop
$ifIndex = (Get-NetAdapter -InterfaceDescription "Hyper-V Virtual Ethernet*" | Sort-Object
 | Select ifIndex).ifIndex
New-NetRoute -DestinationPrefix 169.254.170.2/32 -InterfaceIndex $ifIndex -NextHop $gateway
```

将 Windows 映像推送到 Amazon ECR

您可以将 Windows Docker 容器映像推送到 Amazon ECR。您必须使用支持 Windows 容器的 Docker 版本。以下过程介绍如何拉取 Windows Docker 映像，创建 Amazon ECR 存储库来存储映像，为映像添加存储库标签，为 Amazon ECR 注册表验证映像，然后将映像推送到存储库。

要拉取 Windows Docker 映像并添加标签

1. 将 Windows Docker 映像拉取到本地。本示例使用 `microsoft/iis` 映像。

```
PS C:\> docker pull microsoft/iis
Using default tag: latest
latest: Pulling from microsoft/iis

3889bb8d808b: Pull complete
04ee5d718c7a: Pull complete
c0931dd15237: Pull complete
61784b745c20: Pull complete
d05122f129ca: Pull complete
Digest: sha256:25586570b058da9882d4af640d326d0cc26dfd60b67e1cee63f35ea54d83c882
Status: Downloaded newer image for microsoft/iis:latest
```

2. 为您的映像创建 Amazon ECR 存储库。

```
PS C:\> aws ecr create-repository --repository-name iis
{
    "repository": {
        "registryId": "111122223333",
        "repositoryName": "iis",
        "repositoryArn": "arn:aws:ecr:us-west-2:111122223333:repository/iis",
        "createdAt": 1481845593.0,
        "repositoryUri": "111122223333.dkr.ecr.us-west-2.amazonaws.com/iis"
    }
}
```

3. 使用从上一条命令返回的 `repositoryUri` 为映像添加标签。

```
PS C:\> docker tag microsoft/iis 111122223333.dkr.ecr.us-west-2.amazonaws.com/iis
```

4. 将您的 Docker 客户端验证到 Amazon ECR 注册表。

   Note

   AWS CLI 从版本 1.9.15 开始提供 get-login 命令；但对于较新的 Docker 版本（17.06 或更高版本），我们建议使用 1.11.91 版或更高版本。可以使用 aws --version 命令查看 AWS CLI 版本。如果您使用的是 Docker 17.06 或更高版本，请在 `get-login` 后包含 `--no-include-email` 选项。如果收到 `Unknown options: --no-include-email` 错误，请安装最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

```
PS C:\> Invoke-Expression -Command (aws ecr get-login)
```

5. 将映像推送到 Amazon ECR。

```
PS C:\> docker push 111122223333.dkr.ecr.us-west-2.amazonaws.com/iis
The push refers to a repository [111122223333.dkr.ecr.us-west-2.amazonaws.com/iis]
1e4f77a75bd4: Pushed
ac90fb7da567: Pushed
c7090349c7b3: Pushed
b9454c3094c6: Skipped foreign layer
3fd27ecef6a3: Skipped foreign layer
latest: digest: sha256:0ddc7af8691072bb2dd8b3f189388b33604c90774d3dc0485b1bf379f9bec4c5
 size: 1574
```

文档历史记录

下表描述了 Amazon Elastic Container Service Developer Guide 的主要更新和新功能。我们还经常更新文档来处理您发送给我们的反馈意见。

变更	描述	日期
弹性网络接口中继	增加了针对 ENI 中继功能的其他受支持的 Amazon EC2 实例类型。有关更多信息，请参阅 支持的 Amazon EC2 实例类型 (p. 178)。	2019 年 8 月 1 日
向服务注册多个目标组	增加了对在服务定义中指定多个目标组的支持。有关更多信息，请参阅 向服务注册多个目标组 (p. 298)。	2019 年 7 月 30 日
使用 Secrets Manager 密钥指定敏感数据	增加了有关使用 Secrets Manager 密钥指定敏感数据的教程。有关更多信息，请参阅 教程：使用 Secrets Manager 密钥指定敏感数据 (p. 557)。	2019 年 7 月 20 日
Amazon ECS CLI v1.15.0	已发布新版本的 Amazon ECS CLI。有关更多信息，请参阅 Amazon ECS CLI 更改日志。	2019 年 7 月 9 日
CloudWatch Container Insights	Amazon ECS 增加了对 CloudWatch Container Insights 的支持。有关更多信息，请参阅Amazon ECS CloudWatch Container Insights (p. 363)。	2019 年 7 月 9 日
Amazon ECS 服务和任务集的资源级权限	Amazon ECS 扩展了对 Amazon ECS 服务和任务的资源级权限支持。有关更多信息，请参阅 Amazon Elastic Container Service 如何与 IAM 协同工作 (p. 370)。	2019 年 6 月 27 日
针对 AWS-2019-005 进行修补的新的 Amazon ECS 优化 AMI	Amazon ECS 已更新了经 Amazon ECS 优化的 AMI，解决了 AWS-2019-005 中描述的漏洞。	2019 年 6 月 17 日
弹性网络接口中继	Amazon ECS 引入了对使用已增加弹性网络接口 (ENI) 密度的受支持的 Amazon EC2 实例类型来启动容器实例的支持。通过使用这些实例类型并选择使用 `awsvpcTrunking` 账户设置，可以在新启动的容器实例上提供更大的 ENI 密度，从而允许您在每个容器实例上放置更多任务。有关更多信息，请参阅 弹性网络接口中继 (p. 175)。	2019 年 6 月 6 日
AWS Fargate 平台版本 1.3.0 更新	从 2019 年 5 月 1 日开始，所启动的任何新的 Fargate 任务均支持 `splunk` 日志驱动程序以及 `awslogs` 日志驱动程序。有关更多信息，请参阅存储和日志记录 (p. 63)。	2019 年 5 月 1 日
AWS Fargate 平台版本 1.3.0 更新	从 2019 年 5 月 1 日开始，所启动的任何新的 Fargate 任务均支持使用 `secretOptions` 容器定义参数来引用容器的日志配置中的敏感数据。有关更多信息，请参阅指定敏感数据 (p. 107)。	2019 年 5 月 1 日
AWS Fargate 平台版本 1.3.0 更新	从 2019 年 4 月 2 日开始，所启动的任何新的 Fargate 任务均支持向容器中注入敏感数据，方式是将您的敏感数据存储在 AWS Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数中，然后在容器定义中引用它们。有关更多信息，请参阅指定敏感数据 (p. 107)。	2019 年 4 月 2 日

变更	描述	日期
AWS Fargate 平台版本 1.3.0 更新	从 2019 年 3 月 27 日开始，启动的任何新的 Fargate 任务均可以使用其他任务定义参数，您可以通过这些参数定义代理配置、容器启动和关闭的依赖条件，以及每个容器的启动和停止超时值。有关更多信息，请参阅代理配置 (p. 79)、容器依赖项 (p. 71)和容器超时 (p. 72)。	2019 年 3 月 27 日
Amazon ECS 引入了外部部署类型	外部 部署类型使您能够使用任何第三方部署控制器来完全控制 Amazon ECS 服务的部署过程。有关更多信息，请参阅外部部署 (p. 281)。	2019 年 3 月 27 日
AWS App Mesh 和 Amazon ECS 入门	增加了有关 AWS App Mesh 和 Amazon ECS 的教程。有关更多信息，请参阅 AWS App Mesh 和 Amazon ECS 入门 (p. 544)。	2019 年 3 月 27 日
Amazon ECS 上的 AWS Deep Learning Containers	AWS Deep Learning Containers 是一组 Docker 映像，用于在 Amazon Elastic Container Service (Amazon ECS) 上的 TensorFlow 中训练和处理模型。Deep Learning Containers 为优化环境提供了 TensorFlow、Nvidia CUDA（适用于 GPU 实例）和 Intel MKL（适用于 CPU 实例）库且在 Amazon ECR 中可用。有关更多信息，请参阅 Amazon ECS 上的 AWS Deep Learning Containers (p. 554)。	2019 年 3 月 27 日
Amazon ECS 引入了增强型容器依赖项管理	Amazon ECS 引入了其他任务定义参数，使您能够定义容器启动和关闭的依赖项，以及每个容器的启动和停止超时值。有关更多信息，请参阅容器依赖项 (p. 71)。	2019 年 3 月 7 日
Amazon ECS CLI v1.13.0	已发布新版本的 Amazon ECS CLI。有关更多信息，请参阅 Amazon ECS CLI 更改日志。	2019 年 3 月 7 日
Amazon ECS 引入了 `PutAccountSettingDefault` API	Amazon ECS 引入了 `PutAccountSettingDefault` API，该 API 允许用户为账户上的所有 IAM 用户和角色设置默认 ARN/ID 格式加入状态。以前，设置账户的默认加入状态需要使用根用户。 有关更多信息，请参阅Amazon 资源名称 (ARN) 和 ID (p. 121)。	2019 年 2 月 8 日
Amazon ECS 支持 GPU 工作负载	Amazon ECS 通过使您能够使用支持 GPU 的容器实例创建集群，引入了对 GPU 工作负载的支持。在任务定义中，您可以指定所需 GPU 的数量，而且 ECS 代理会将物理 GPU 固定到容器中。 有关更多信息，请参阅在 Amazon ECS 上使用 GPU (p. 35)。	2019 年 2 月 4 日
Amazon ECS 扩展密钥支持	Amazon ECS 扩展了以下支持：可直接在您的任务定义中使用 AWS Secrets Manager 密钥将敏感数据注入您的容器。 有关更多信息，请参阅指定敏感数据 (p. 107)。	2019 年 1 月 21 日
接口 VPC 终端节点 (AWS PrivateLink)	添加了对配置由 AWS PrivateLink 提供支持的接口 VPC 终端节点的支持。这能让您在您的 VPC 和 Amazon ECS 之间创建私有连接，而无需通过 Internet、NAT 实例、VPN 连接或 AWS Direct Connect 进行访问。 有关更多信息，请参阅推理 VPC 终端节点 (AWS PrivateLink)。	2018 年 12 月 26 日

变更	描述	日期
AWS Fargate 平台版本 1.3.0	新 AWS Fargate 平台版本已发布，其中包含： • 增加了对使用 AWS Systems Manager Parameter Store 参数向容器中注入敏感数据的支持。 有关更多信息，请参阅指定敏感数据 (p. 107)。 • 增加了 Fargate 任务的任务回收，此过程用于刷新作为 Amazon ECS 服务一部分的任务。 有关更多信息，请参阅Fargate 任务回收 (p. 268)。 有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。	2018 年 12 月 17 日
更新了服务限制	更新了以下服务限制： • 每个区域每个账户的集群数从 `1000` 提高到 `2000`。 • 每个集群的容器实例数从 `1000` 提高到 `2000`。 • 每个集群的服务数从 `500` 提高到 `1000`。 有关更多信息，请参阅Amazon ECS 服务限制 (p. 540)。	2018 年 12 月 14 日
AWS Fargate 区域扩展	带 Amazon ECS 的 AWS Fargate 已扩展到亚太地区（孟买）和加拿大（中部）区域。 有关更多信息，请参阅 AWS Fargate 平台版本 (p. 33)。	2018 年 12 月 7 日
Amazon ECS 蓝/绿部署	Amazon ECS 增加了对使用 CodeDeploy 进行蓝/绿部署的支持。此部署类型允许您先验证服务的新部署，然后再向其发送生产流量。 有关更多信息，请参阅使用 CodeDeploy 进行蓝/绿部署 (p. 278)。	2018 年 11 月 27 日
发布了Amazon ECS-optimized Amazon Linux 2 (arm64) AMI	Amazon ECS 发布了适用于 arm64 架构的Amazon ECS-optimized Amazon Linux 2 AMI。 有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2018 年 11 月 26 日
Amazon ECS CLI v1.11.2	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了对使用 AWS Systems Manager Parameter Store 参数向容器中注入敏感数据的支持。有关更多信息，请参阅使用 Amazon ECS 参数 (p. 520)。 • 增加了对在任务定义中指定 `ipcMode` 和 `pidMode` Docker 标志的支持。有关更多信息，请参阅使用 Amazon ECS 参数 (p. 520)。	2018 年 11 月 19 日
增加了对任务定义中的其他 Docker 标志的支持	Amazon ECS 在任务定义中引入了对以下 Docker 标志的支持： • IPC 模式 (p. 79) • PID 模式 (p. 79)	2018 年 11 月 16 日

变更	描述	日期
Amazon ECS 密钥支持	Amazon ECS 增加了对使用 AWS Systems Manager Parameter Store 参数向容器中注入敏感数据的支持。 有关更多信息，请参阅指定敏感数据 (p. 107)。	2018 年 11 月 15 日
为资源添加标签	Amazon ECS 增加了对将元数据标签添加到您的服务、任务定义、任务、集群和容器实例的支持。 有关更多信息，请参阅资源和标签 (p. 330)。	2018 年 11 月 15 日
AWS Fargate 区域扩展	带 Amazon ECS 的 AWS Fargate 已扩展到美国西部（加利福利亚北部）和亚太地区（首尔）区域。 有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。	2018 年 11 月 7 日
更新了服务限制	更新了以下服务限制： • 每个区域每个账户对应的使用 Fargate 启动类型的任务数从 20 提高到 50。 • 使用 Fargate 启动类型的任务的公有 IP 地址数从 20 提高到 50。 有关更多信息，请参阅Amazon ECS 服务限制 (p. 540)。	2018 年 10 月 31 日
AWS Fargate 区域扩展	带 Amazon ECS 的 AWS Fargate 已扩展到欧洲（伦敦）区域。 有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。	2018 年 10 月 26 日
Amazon ECS CLI v1.10.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了 ecs-cli registry-creds 命令，它有助于在 Amazon ECS 中创建和使用私有注册表凭证。有关更多信息，请参阅ecs-cli registry-creds (p. 506)。 • 添加了对 Amazon Linux 2 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2018 年 10 月 25 日
发布了Amazon ECS-optimized Amazon Linux 2 AMI	Amazon ECS 提供了已针对服务进行了优化的 Linux AMI，具有两种变体。最新的推荐版本基于 x;。Amazon ECS 还提供了基于 Amazon Linux AMI 的 AMI，但我们建议您将工作负载迁移到 Amazon Linux 2 变体，因为对 Amazon Linux AMI 的支持将截止于 2020 年 6 月 30 日。 有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2018 年 10 月 18 日
Amazon ECS CLI v1.9.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 添加了对 服务发现 的支持。有关更多信息，请参阅教程：使用 Amazon ECS CLI 创建使用服务发现的 Amazon ECS 服务 (p. 442)。 • 增加了对 Amazon ECS 集群中的 Amazon EC2 Spot 实例的支持。 • 增加了对自定义用户数据的支持。	2018 年 10 月 18 日

变更	描述	日期
Amazon ECS 任务元数据终端节点版本 3	从 Amazon ECS 容器代理版本 1.21.0 开始，代理将称为 `ECS_CONTAINER_METADATA_URI` 的环境变量注入任务中的每个容器。在您查询任务元数据版本 3 终端节点时，各种任务元数据和 Docker 统计数据都可用于那些在 Amazon ECS 容器代理所提供的 HTTP 终端节点上使用 `awsvpc` 网络模式的任务。有关更多信息，请参阅 Amazon ECS 任务元数据终端节点 (p. 236)。	2018 年 10 月 18 日
Amazon ECS 服务发现区域扩展	Amazon ECS 服务发现已将支持扩展到加拿大（中部）、南美洲（圣保罗）、亚太地区（首尔）、亚太地区（孟买）和欧洲（巴黎）区域。 有关更多信息，请参阅 服务发现 (p. 306)。	2018 年 9 月 27 日
增加了对容器定义中对其他 Docker 标志的支持。	Amazon ECS 在容器定义中引入了对以下 Docker 标志的支持： • 系统控制 (p. 73) • 交互式 (p. 74) • 伪终端 (p. 74)	2018 年 9 月 17 日
Amazon ECS 使用 AWS Fargate 任务对私有注册表身份验证的支持。	Amazon ECS 对 Fargate 任务引入了使用 AWS Secrets Manager 进行私有注册表身份验证的支持。此功能使您能够在容器定义中安全地存储并随后引用您的凭证，这让您的任务可以使用私有映像。 有关更多信息，请参阅任务的私有注册表身份验证 (p. 104)。	2018 年 9 月 10 日
Amazon ECS CLI v1.8.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了对 Docker 复合文件中的 Docker 卷的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。 • 添加了对 Docker 复合文件中的任务放置约束和策略的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。 • 增加了对 Docker 复合文件中的私有注册表身份验证的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。 • 增加了对 `compose up` 上 `--force-update` 的支持以强制重新启动任务。有关更多信息，请参阅ecs-cli compose up (p. 480)。	2018 年 9 月 7 日
Amazon ECS 服务发现区域扩展	Amazon ECS 服务发现已将支持扩展到亚太地区（新加坡）、亚太地区（悉尼）、亚太地区（东京）、欧洲（法兰克福）和欧洲（伦敦）区域。 有关更多信息，请参阅 服务发现 (p. 306)。	2018 年 8 月 30 日
计划的任务与 Fargate 任务支持	Amazon ECS 为 Fargate 启动类型引入了对计划的任务的支持。 有关更多信息，请参阅排定的任务 (`cron`) (p. 263)。	2018 年 8 月 28 日

变更	描述	日期
使用 AWS Secrets Manager 支持的私有注册表身份验证	Amazon ECS 引入了使用 AWS Secrets Manager 对私有注册表身份验证的支持。此功能使您能够在容器定义中安全地存储并随后引用您的凭证，这让您的任务可以使用私有映像。 有关更多信息，请参阅 任务的私有注册表身份验证 (p. 104)。	2018 年 8 月 16 日
添加了 Docker 卷支持	Amazon ECS 引入了对 Docker 卷的支持。 有关更多信息，请参阅 在任务中使用数据卷 (p. 83)。	2018 年 8 月 9 日
AWS Fargate 区域扩展	带 Amazon ECS 的 AWS Fargate 已扩展到欧洲（法兰克福）、亚太地区（新加坡）和亚太地区（悉尼）区域。 有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。	2018 年 7 月 19 日
Amazon ECS CLI v1.7.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了对 Docker 复合文件中的容器 `healthcheck` 和 `devices` 的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。	2018 年 7 月 18 日
添加了 Amazon ECS 服务计划程序策略	Amazon ECS 引入了服务计划程序策略的概念。 有两种服务计划程序策略可用： • `REPLICA` - 副本计划策略在群集上放置并维护所需数量的任务。默认情况下，服务计划程序可在多个可用区之间分布任务。您可以使用任务放置策略和约束自定义任务放置决策。有关更多信息，请参阅 副本 (p. 271)。 • `DAEMON` - 守护程序计划策略只在每个活动容器实例上部署一个任务，以满足您在群集中指定的所有任务放置约束。当使用此策略时，无需指定所需的任务数、任务放置策略，也无需使用服务 Auto Scaling 策略。有关更多信息，请参阅 守护程序 (p. 270)。 Note Fargate 任务不支持 `DAEMON` 计划策略。 有关更多信息，请参阅 服务计划程序概念 (p. 270)。	2018 年 6 月 12 日
Amazon ECSCLI v1.6.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 添加了对 Docker 复合文件语法版本 3 的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。	2018 年 6 月 5 日

变更	描述	日期
Amazon ECS 容器代理 1.18.0 版	发布的新版本的 Amazon ECS 容器代理，其中增加了以下功能： • 添加了在非 Amazon Linux EC2 实例上手动安装来自 S3 URL 的容器代理的过程，包括用于验证 Amazon ECS 容器代理安装文件的 PGP 签名方法。有关更多信息，请参阅安装 Amazon ECS 容器代理 (p. 198)。 • 添加了在非 Windows EC2 实例上手动安装来自 S3 URL 的容器代理的过程，包括用于验证 Amazon ECS 容器代理安装文件的 PGP 签名方法。有关更多信息，请参阅 Windows 容器入门 (p. 600)。 • 添加了对使用 `ECS_IMAGE_PULL_BEHAVIOR` 参数自定义容器代理映像拉取行为的支持。有关更多信息，请参阅Amazon ECS 容器代理配置 (p. 217)。 有关更多信息，请参阅 amazon-ecs-agent github。	2018 年 5 月 24 日
添加了在配置 服务发现 时对 `bridge` 和 `host` 网络模式的支持	添加了对使用指定 `bridge` 或 `host` 网络模式的任务定义来为 Amazon ECS 服务配置 服务发现 的支持。有关更多信息，请参阅服务发现 (p. 306)。	2018 年 5 月 22 日
增加了对其他Amazon ECS-optimized AMI 元数据参数的支持	增加了子参数，允许您以编程方式检索Amazon ECS-optimized AMI ID、映像名称、操作系统、容器代理版本和运行时版本。使用 Systems Manager Parameter Store API 查询元数据。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。	2018 年 5 月 9 日
AWS Fargate 区域扩展	带 Amazon ECS 的 AWS Fargate 已扩展到美国东部（俄亥俄）、美国西部（俄勒冈）和欧洲西部（爱尔兰）区域。 有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate (p. 28)。	2018 年 4 月 26 日
Amazon ECSCLI v1.5.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了对 ECS CLI 自动检索最新稳定的Amazon ECS-optimized AMI（方法为在创建集群资源过程中查询 Systems Manager Parameter Store API）的支持。这要求您正在使用的用户账户具有所需的 Systems Manager 权限。有关更多信息，请参阅 ecs-cli up (p. 453)。 • 添加了针对复合文件中 `shm_size` 和 `tmpfs` 参数的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。 有关更新后的 ECS CLI 语法的更多信息，请参阅 Amazon ECS 命令行参考 (p. 445)。	2018 年 4 月 19 日
Amazon ECS-optimized AMI 元数据检索	增加了使用 Systems Manager Parameter Store API 以编程方式检索Amazon ECS-optimized AMI 元数据的功能。有关更多信息，请参阅检索 Amazon ECS-Optimized AMI 元数据 (p. 155)。	2018 年 4 月 10 日
Amazon ECS CLI 下载验证	添加了新的 PGP 签名方法以验证 Amazon ECS CLI 安装文件。有关更多信息，请参阅 安装 Amazon ECS CLI (p. 425)。	2018 年 4 月 5 日

变更	描述	日期
AWS Fargate 平台版本	新 AWS Fargate 平台版本已发布，其中包含： • 添加了对 Amazon ECS 任务元数据终端节点 (p. 236) 的支持。 • 添加了对 运行状况检查 (p. 57) 的支持。 • 添加了对 服务发现 (p. 306) 的支持 有关更多信息，请参阅AWS Fargate 平台版本 (p. 33)。	2018 年 3 月 26 日
Amazon ECS 服务发现	增加了与 Route 53 的集成，以支持 Amazon ECS 服务发现。有关更多信息，请参阅服务发现 (p. 306)。	2018 年 3 月 22 日
Amazon ECS CLI v1.4.2	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 将 AMI 更新为 `amzn-ami-2017.09.k-amazon-ecs-optimized`。 有关更新后的 ECS CLI 语法的更多信息，请参阅 Amazon ECS 命令行参考 (p. 445)。	2018 年 3 月 20 日
Docker shm-size 和 tmpfs 支持	添加了在 Amazon ECS 任务定义中对 Docker shm-size 和 tmpfs 参数的支持。 有关更新后的 ECS CLI 语法的更多信息，请参阅 Linux 参数 (p. 69)。	2018 年 3 月 20 日
Amazon ECS CLI v1.4.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 增加了对 us-gov-west-1 区域的支持。 • 为 compose service 命令添加了 `--force-deployment` 标志。有关更多信息，请参阅ecs-cli compose service (p. 482)。 • 增加了对 ECS 配置文件中 `aws_session_token` 的支持。有关更多信息，请参阅 ecs-cli configure profile (p. 449)。 • 将 AMI 更新为 `amzn-ami-2017.09.j-amazon-ecs-optimized`。 有关更新后的 ECS CLI 语法的更多信息，请参阅 Amazon ECS 命令行参考 (p. 445)。	2018 年 3 月 09 日
容器运行状况检查	增加了对容器定义中 Docker 运行状况检查的支持。有关更多信息，请参阅运行状况检查 (p. 57)。	2018 年 3 月 08 日
AWS Fargate	增加了有关使用 AWS Fargate 的 Amazon ECS 的概览。有关更多信息，请参阅Amazon ECS 上的 AWS Fargate (p. 28)。	2018 年 2 月 22 日
Amazon ECS 任务元数据终端节点	从 1.17.0 版 Amazon ECS 容器代理开始，各种任务元数据和 Docker 统计数据都可用于那些在 Amazon ECS 容器代理所提供的 HTTP 终端节点上使用 `awsvpc` 网络模式的任务。有关更多信息，请参阅Amazon ECS 任务元数据终端节点 (p. 236)。	2018 年 2 月 8 日

变更	描述	日期
使用目标跟踪策略的 Amazon ECS 服务 Auto Scaling	增加了在 Amazon ECS 控制台中对使用目标跟踪策略的 ECS 服务 Auto Scaling 的支持。有关更多信息，请参阅 目标跟踪扩展策略 (p. 300)。 删除了 ECS 首次运行向导中步进扩展以前的教程。使用目标跟踪的新教程进行了替换。	2018 年 2 月 8 日
Amazon ECSCLI v1.3.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 能够使用 up 命令创建空集群。 • 为 compose service up 命令添加了 `--health-check-grace-period` 标志。 • 将 AMI 更新为 `amzn-ami-2017.09.g-amazon-ecs-optimized`。 有关更新后的 ECS CLI 语法的更多信息，请参阅 Amazon ECS 命令行参考 (p. 445)。	2018 年 1 月 19 日
Docker 17.09 支持	添加对 Docker 17.09 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2018 年 1 月 18 日
Elastic Load Balancing 运行状况检查初始化等待期	增加了为运行状况检查指定等待期的功能。	2017 年 12 月 27 日
新服务计划程序行为	更新了有关无法启动的服务任务的行为的信息。记录了在服务任务连续失败时触发的新服务事件消息。有关此更新的行为的更多信息，请参阅附加服务概念 (p. 271)。	2018 年 1 月 11 日
任务级 CPU 和内存	增加了对在任务定义中指定任务级 CPU 和内存的支持。有关更多信息，请参阅 TaskDefinition。	2017 年 12 月 12 日
Amazon ECS 控制台 CodePipeline 集成	增加了 Amazon ECS 与 CodePipeline 的集成。CodePipeline 支持 Amazon ECS 作为部署选项来帮助设置部署管道。有关更多信息，请参阅教程：CodePipeline 持续部署 (p. 569)。	2017 年 12 月 12 日
任务执行角色	Amazon ECS 容器代理将代表您调用 Amazon ECS API 操作，因此，容器实例需要服务的 IAM 策略和角色，以便了解属于您的代理。任务执行角色涵盖以下操作： • 调用 Amazon ECR 以拉取容器映像 • 调用 CloudWatch 以存储容器应用程序日志 有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 (p. 406)。	2017 年 12 月 7 日
Windows 容器支持 GA	增加了对 Windows 2016 容器的支持。有关更多信息，请参阅Windows 容器 (p. 599)。	2017 年 12 月 5 日

变更	描述	日期
带有 Fargate 支持的 Amazon ECS CLI v1.1.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 支持任务联网 • 支持 AWS Fargate • 支持从任务查看 CloudWatch Logs 数据 有关更多信息，请参阅 ECS CLI 更改日志。	2017 年 11 月 29 日
AWS Fargate GA	增加了对使用 Fargate 启动类型启动 Amazon ECS 服务的支持。有关更多信息，请参阅Amazon ECS 启动类型 (p. 81)。	2017 年 11 月 29 日
Amazon ECS 名称变更	重命名了 Amazon Elastic Container Service (以前名为 Amazon EC2 Container Service)。	2017 年 11 月 21 日
任务联网	awsvpc 网络模式提供的任务联网功能使 Amazon ECS 任务具有与 Amazon EC2 实例相同的联网属性。当您在任务定义中使用 awsvpc 网络模式时，每个从该任务定义启动的任务都会获取其自己的弹性网络接口、主要私有 IP 地址和内部 DNS 主机名。任务联网功能简化了容器联网，使您可以更好地控制容器化应用程序如何与您的 VPC 内的其他服务进行通信。有关更多信息，请参阅 `awsvpc` 网络模式的任务联网 (p. 95)。	2017 年 11 月 14 日
Amazon ECSCLI v1.0.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 支持添加多个命名配置文件和集群配置 • 支持使用 `--ecs-params` 指定的自定义任务定义参数 • 支持在 Windows 上运行 Amazon ECS CLI 有关更多信息，请参阅 ECS CLI 更改日志。	2017 年 11 月 7 日
Amazon ECS 容器元数据	Amazon ECS 容器现在能够访问元数据，如它们的 Docker 容器或映像 ID、网络配置或 Amazon ARN。有关更多信息，请参阅Amazon ECS 容器元数据文件 (p. 232)。	2017 年 11 月 2 日
Docker 17.06 支持	添加对 Docker 17.06 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2017 年 11 月 2 日
支持 Docker 标志：device 和 init	增加了对任务定义中使用 `LinuxParameters` 参数 (`devices` 和 `initProcessEnabled`) 的 Docker 的 device 和 init 功能的支持。有关更多信息，请参阅 LinuxParameters。	2017 年 11 月 2 日
支持 Docker 标志：cap-add 和 cap-drop	增加了对任务定义中使用 `LinuxParameters` 参数 (`capabilities`) 的 Docker 的 cap-add 和 cap-drop 功能的支持。有关更多信息，请参阅 LinuxParameters。	2017 年 9 月 22 日
网络负载均衡器支持	Amazon ECS 在 Amazon ECS 控制台中增加了对 网络负载均衡器 的支持。有关更多信息，请参阅创建 网络负载均衡器 (p. 293)。	2017 年 9 月 7 日
RunTask 覆盖	增加了对运行任务时执行任务定义覆盖的支持。这允许您在运行任务的同时更改任务定义，而无需创建新的任务定义修订。有关更多信息，请参阅 正在运行的任务 (p. 251)。	2017 年 6 月 27 日

变更	描述	日期
Amazon ECS 计划任务	增加了对使用 cron 计划任务的支持。有关更多信息，请参阅 排定的任务 (cron) (p. 263)。	2017 年 6 月 7 日
Amazon ECS 控制台中的 Spot 实例	增加了对在 Amazon ECS 控制台中创建 Spot 队列容器实例的支持。有关更多信息，请参阅 启动 Amazon ECS 容器实例 (p. 165)。	2017 年 6 月 6 日
Amazon ECS CLI v0.5.0	发布的新版本的 Amazon ECS CLI，其中增加了以下功能： • 能够推送、提取和列出 Amazon ECR 映像 • 支持 CreateService 中的现有负载均衡器和应用程序负载均衡器 有关更多信息，请参阅 ECS CLI 更改日志。	2017 年 4 月 3 日
新的Amazon ECS-optimized AMI 版本的 Amazon SNS 通知	增加了订阅有关新的Amazon ECS-optimized AMI 版本的 SNS 通知的功能。有关更多信息，请参阅订阅Amazon ECS-Optimized Amazon Linux AMI 更新通知 (p. 161)。	2017 年 3 月 23 日
微服务和批处理作业	增加了 Amazon ECS 的两个常见使用案例 (微服务和批处理作业) 的文档。有关更多信息，请参阅 Amazon ECS 中的常见使用案例 (p. 541)。	2017 年 2 月
容器实例耗尽	增加了对容器实例耗尽的支持，这提供了一种从集群中删除容器实例的方法。有关更多信息，请参阅容器实例耗尽 (p. 190)。	2017 年 1 月 24 日
Docker 1.12 支持	添加对 Docker 1.12 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2017 年 1 月 24 日
新任务放置策略	增加了对任务放置策略的支持：基于属性的放置、装填、可用区分散以及每个主机一个。有关更多信息，请参阅 Amazon ECS 任务放置策略 (p. 255)。	2016 年 12 月 29 日
Windows 容器支持测试版	增加了对 Windows 2016 容器的支持 (测试版)。有关更多信息，请参阅 Windows 容器 (p. 599)。	2016 年 12 月 20 日
Blox OSS 支持	增加了对 Blox OSS 的支持，从而实现了自定义任务计划程序。有关更多信息，请参阅计划 Amazon ECS 任务 (p. 250)。	2016 年 12 月 1 日
CloudWatch Events 的 Amazon ECS 事件流	Amazon ECS 现在将容器实例和任务状态更改发送到 CloudWatch Events。有关更多信息，请参阅CloudWatch Events 的 Amazon ECS 事件流 (p. 354)。	2016 年 11 月 21 日
Amazon ECS 容器日志记录到 CloudWatch Logs	增加了对 awslogs 驱动程序将容器日志流发送到 CloudWatch Logs 的支持。有关更多信息，请参阅使用 awslogs 日志驱动程序 (p. 97)。	2016 年 12 月 9 日
带有支持动态端口的 Elastic Load Balancing 的 Amazon ECS 服务	增加了对负载均衡器支持每个侦听器多个实例:端口组合的支持，这会增加容器的灵活性。现在，您可以让 Docker 动态定义容器的主机端口，而 ECS 计划程序向负载均衡器注册实例:端口。有关更多信息，请参阅服务负载均衡 (p. 284)。	2016 年 8 月 11 日
Amazon ECS 任务的 IAM 角色	增加了对将 IAM 角色与任务关联的支持。这为容器提供了更精细的权限，而不是对整个容器实例使用单个角色。有关更多信息，请参阅 任务的 IAM 角色 (p. 418)。	2016 年 7 月 13 日

变更	描述	日期
Amazon ECS CLI 支持 Docker Compose v2 格式	Amazon ECS CLI 增加了对 Docker Compose v2 格式的支持。有关更多信息，请参阅ecs-cli compose (p. 472)。	2016 年 7 月 8 日
Docker 1.11 支持	添加对 Docker 1.11 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2016 年 31 月 5 日
任务自动扩展	Amazon ECS 增加了对自动扩展服务所运行的任务的支持。有关更多信息，请参阅 服务 Auto Scaling (p. 299)。	2016 年 18 月 5 日
基于任务系列的任务定义筛选	增加了对基于任务定义系列筛选任务定义列表的支持。有关更多信息，请参阅 ListTaskDefinitions。	2016 年 5 月 17 日
Docker 容器和 Amazon ECS 代理日志记录	Amazon ECS 增加了将 ECS 代理和 Docker 容器日志从容器实例发送到 CloudWatch Logs 以简化问题排查的功能。	2016 年 5 月 5 日
发布了 Amazon ECS CLI v0.3	发布了新版本的 Amazon ECS CLI，它增加了对使用负载均衡器创建服务的支持。	2016 年 4 月 11 日
经 ECS 优化的 AMI 现在支持 Amazon Linux 2016.03。	经 ECS 优化的 AMI 增加了对 Amazon Linux 2016.03 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2016 年 4 月 5 日
Docker 1.9 支持	添加对 Docker 1.9 的支持。有关更多信息，请参阅 Amazon ECS-optimized AMI (p. 127)。	2015 年 12 月 22 日
有关集群 CPU 和内存预留的 CloudWatch 指标	Amazon ECS 增加了有关 CPU 和内存预留的自定义 CloudWatch 指标。	2015 年 12 月 22 日
Amazon ECR	向控制台中添加了新的 Amazon ECR 服务，这增加了对存储与 Docker Hub 或 IAM 用户关联的资源级权限所控制的映像的支持。可在所有 AWS 区域中使用，将自动复制并全球缓存映像，以便启动数百个容器与启动单个容器一样快。	2015 年 12 月 21 日
新的 Amazon ECS 首次运行体验	Amazon ECS 控制台首次运行体验增加了零单击角色创建。	2015 年 11 月 23 日
跨可用区放置任务	Amazon ECS 服务计划程序增加了对跨可用区放置任务的支持。	2015 年 10 月 8 日
支持 Docker Compose 的 Amazon ECS CLI	Amazon ECS CLI 增加了对 Docker Compose 的支持。	2015 年 10 月 8 日
Amazon ECS 集群和服务的 CloudWatch 指标	Amazon ECS 为集群中的每个容器实例、服务和任务定义系列增加了有关 CPU 和内存使用率的自定义 CloudWatch 指标。这些新指标可用于使用 Auto Scaling 组扩展集群中的容器实例或创建自定义 CloudWatch 警报。	2015 年 8 月 17 日
UDP 端口支持	增加了对任务定义中的 UDP 端口的支持。	2015 年 7 月 7 日
环境变量覆盖	增加了对 runTask 的 deregisterTaskDefinition 和环境变量覆盖的支持。	2015 年 6 月 18 日
自动化 Amazon ECS 代理更新	增加了查看容器实例上运行的 ECS 代理版本的功能。还能够从 AWS 管理控制台、AWS CLI 和开发工具包更新 ECS 代理。	2015 年 6 月 11 日

变更	描述	日期
Amazon ECS 服务计划程序和 Elastic Load Balancing 集成	增加了定义服务并将该服务与 Elastic Load Balancing 负载均衡器关联的功能。	2015 年 4 月 9 日
Amazon ECS GA	Amazon ECS 在美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、亚太地区（东京）和欧洲（爱尔兰）区域中正式发布。	2015 年 4 月 9 日

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 AWS 词汇表。