aws

开发人员指南

Amazon Elastic Container Service

Amazon Elastic Container Service: 开发人员指南

Table of Contents

什么是 Amazon ECS？ 1
Amazon ECS 术语和组件 1
Amazon ECS 容量 2
Amazon ECS 控制器 3
Amazon ECS 预调配 3
应用程序生命周期 3
相关信息 5
开始使用 7
设置 7
注册 AWS 账户 7
创建具有管理访问权限的用户 8
创建 Virtual Private Cloud 9
创建安全组 10
创建凭证以连接到 EC2 实例 12
安装 AWS CLI 13
创建容器镜像 13
先决条件 13
创建 Docker 映像 15
将映像推送到 Amazon Elastic Container Registry 17
清理 19
后续步骤 19
了解如何创建 Fargate 启动类型的 Linux 任务 19
先决条件 19
第 1 步：创建集群 20
第 2 步：创建任务定义 21
第 3 步：创建服务 22
步骤 4：查看您的服务 22
第 5 步：清理 23
了解如何创建 Fargate 启动类型的 Windows 任务 23
先决条件 23
步骤 1：创建集群 24
步骤 2：注册 Windows 任务定义 25
步骤 3：使用您的任务定义创建服务 26
步骤 4：查看您的服务 27

第 5 步：清除 27
了解如何创建 EC2 启动类型的 Windows 任务 28
先决条件 28
步骤 1：创建集群 29
第 2 步：注册任务定义 30
步骤 3：创建服务 31
步骤 4：查看您的服务 32
第 5 步：清除 32
开发人员工具概述 34
AWS Management Console 34
AWS Command Line Interface 35
AWS CloudFormation 35
AWS Copilot CLI 35
AWS CDK 36
AWS App2Container 36
Amazon ECS CLI 37
与 Amazon ECS 集成的 Docker Desktop 37
AWS 开发工具包 37
Summary 38
使用 AWS Copilot CLI 创建资源 38
安装 AWS Copilot CLI 39
使用 AWS Copilot CLI 部署示例 Amazon ECS 应用程序 47
使用 AWS CDK 48
步骤 1：设置您的 AWS CDK 项目 49
步骤 2：使用 AWS CDK 在 Fargate 上定义容器化 Web 服务器 52
步骤 3：测试 Web 服务器 59
步骤 4：清除 59
后续步骤 59
使用 AWS CloudFormation 创建资源 60
AWS CloudFormation 模板 60
示例模板 61
使用 AWS CLI 从模板创建资源 68
了解有关 AWS CloudFormation 的更多信息 69
开始使用 Amazon ECS CLI 69
安装 Amazon ECS CLI 69
配置 Amazon ECS CLI 77

AWS Fargate 80
演练 80
容量提供程序 80
任务定义 81
平台版本 81
服务负载均衡 81
使用情况指标 82
了解何时使用 Fargate 启动类型的安全注意事项 82
Fargate 安全最佳实践 82
使用 AWS KMS 加密 Fargate 的临时存储 82
使用 Fargate 进行内核系统调用跟踪的 SYS_PTRACE 功能 83
将 Amazon GuardDuty 与 Fargate 运行时监控结合使用 83
Fargate 安全注意事项 84
适用于 Amazon ECS 的 Fargate Linux 平台版本 84
注意事项 85
1.4.0 85
1.3.0 87
迁移到 Linux 平台版本 1.4.0 88
平台版本弃用 88
Fargate 上的 Linux 容器的容器映像拉取行为 90
适用于 Amazon ECS 的 Fargate Windows 平台版本 91
平台版本注意事项 91
1.0.0 92
适用于 Amazon ECS 的 Fargate 上的 Windows 容器注意事项 92
Fargate 上的 Windows 容器的容器映像拉取行为 93
适用于 Amazon ECS 的 Fargate 临时存储 94
Fargate Linux 容器平台版本 94
Fargate Windows 容器平台版本 95
用于 AWS Fargate 临时存储的客户自主管理型密钥 95
Amazon ECS 上的 AWS Fargate 任务维护常见问题 108
什么是 Fargate 任务维护和停用？ 108
任务停用通知中包含哪些内容？ 108
我可以更改任务停用等待时间吗？ 111
我可以通过其他 AWS 服务获取任务停用通知吗？ 111
我可以在任务停用计划后更改任务停用吗？ 112
我可以控制任务替换的时间吗？ 112

Amazon ECS 如何处理属于服务一部分的任务？ 112
Amazon ECS 可以自动处理独立任务吗？ 112
AWS Fargate 区域 112
AWS Fargate 上的 Linux 容器 113
AWS Fargate 上的 Windows 容器 114
为 Amazon ECS 构建您的解决方案 117
容量 117
联网 117
功能访问 118
IAM 角色 119
日志记录 119
启动类型 119
Fargate 119
EC2 122
外部 123
应用程序位于共享子网、本地区域和 Wavelength 区中 123
共享子网 124
Local Zones 125
Wavelength 区域 125
AWS Outposts 上的 Amazon Elastic Container Service 126
注意事项 126
先决条件 126
在 AWS Outposts 上创建集群 126
优化容量和可用性 129
最大程度提高缩放速度 130
处理需求冲击 131
将应用程序连接到互联网 132
公有子网和互联网网关 133
私有子网和 NAT 网关 135
从互联网接收到 Amazon ECS 的入站连接的最佳实践 136
应用程序负载均衡器 136
网络负载均衡器 137
Amazon API Gateway HTTP API 139
通过账户设置访问相关功能 140
Amazon Resource Names (ARN) 和 ID 144
ARN 和资源 ID 格式时间表 146

AWS Fargate 美国联邦信息处理标准（FIPS-140）合规性 146
标记授权 146
标记授权时间表 147
AWS Fargate 任务停用等待时间 148
运行时监控（Amazon GuardDuty 集成） 149
使用控制台查看账户设置 150
修改账户设置 150
恢复到默认账户设置 151
使用 AWS CLI 管理账户设置 151
适用于 Amazon ECS 的 IAM 角色 153
任务定义 156
任务定义状态 157
可以阻止删除的 Amazon ECS 资源 158
设计应用程序架构 158
容器映像的最佳实践 160
任务大小最佳实践 161
网络安全最佳实践 162
EC2 启动类型的任务联网 166
Fargate 启动类型的任务联网 175
任务的存储选项 179
管理容器交换内存空间 251
Fargate 启动类型的任务定义差异 252
运行 Windows 的 EC2 实例的任务定义差异 259
使用控制台创建任务定义 260
JSON 验证 260
AWS CloudFormation 堆栈 260
过程 261
使用控制台更新任务定义 282
JSON 验证 283
过程 283
使用控制台取消注册任务定义修订 284
AWS CloudFormation 堆栈 260
过程 285
使用控制台删除任务定义修订 285
可以阻止删除的 Amazon ECS 资源 158
过程 286

任务定义应用场景 286
适用于 GPU 工作负载的任务定义 287
适用于视频转码工作负载的任务定义 295
适用于 AWS Neuron 机器学习工作负载的任务定义 307
适用于深度学习实例的任务定义 316
适用于 64 位 ARM 工作负载的任务定义 318
将日志发送到 CloudWatch 320
将日志发送到 AWS 服务或 AWS Partner 323
使用非 AWS 容器映像 334
将单个环境变量传递给容器 337
将环境变量传递给容器 338
将敏感数据传递给容器 341
任务定义参数 362
系列 362
启动类型 362
任务角色 363
任务执行角色 363
网络模式 364
运行时平台 365
任务大小 366
容器定义 370
Elastic Inference 加速器名称 411
任务放置约束 411
代理配置 412
卷 414
标签 420
其他任务定义参数 421
任务定义模板 424
示例任务定义 435
Webserver 435
splunk 日志驱动程序 438
fluentd 日志驱动程序 438
gelf 日志驱动程序 439
外部实例上的工作负载 439
Amazon ECR 映像和任务定义 IAM 角色 441
带命令的入口点 441

容器依赖项 442
Windows 示例任务定义 444
集群 446
Fargate 启动类型的集群 447
Fargate Spot 终止通知 448
为 Fargate 启动类型创建集群 450
EC2 启动类型的容量提供程序 452
EC2 容器实例安全性 453
为 Amazon EC2 启动类型创建集群 454
集群 Auto Scaling 458
Amazon EC2 容器实例 481
外部启动类型的集群 619
支持的操作系统和系统体系结构 619
注意事项 620
为外部启动类型创建集群 623
将外部实例注册到 Amazon ECS 集群 625
注销外部实例 630
更新 AWS Systems Manager 代理和 Amazon ECS 容器代理 634
更新集群 639
删除集群 640
创建容量提供程序 641
更新容量提供程序 642
删除容量提供程序 642
注销容器实例 643
过程 644
容器实例耗尽 644
服务的耗尽行为 644
独立任务的耗尽行为 645
过程 646
容器代理 646
生命周期 647
经 Amazon ECS 优化的 AMI 647
其他信息 648
容器代理配置 648
安装 Amazon ECS 容器代理 650
容器代理日志配置参数 656

为私有 Docker 映像配置容器实例 659
清理任务和映像 663
计划您的容器 666
计算选项 667
任务生命周期 668
生命周期状态 669
Amazon ECS 如何将任务放置在容器实例上 670
EC2 启动类型 671
Fargate 启动类型 671
使用策略来定义任务放置 672
与组相关的任务 676
定义哪些容器实例用于任务 677
独立任务 686
任务工作流程 686
优化任务启动时间 686
将应用程序作为任务运行 687
使用 Amazon EventBridge 调度器计划任务 695
停止任务 701
服务 702
进程守护程序策略 703
副本策略 704
服务参数的最佳实践 705
创建服务 708
更新服务 729
更新蓝绿部署 741
删除服务 742
滚动更新部署 743
蓝/绿部署 749
外部部署 768
使用负载均衡分配服务流量 774
服务自动扩展 786
互连服务 795
任务横向缩减保护 841
服务限制逻辑 848
服务定义参数 849
标记资源 879

如何为资源添加标签 879
在创建时标记资源 882
限制 882
Amazon ECS 托管标签 883
使用标签记账 884
将标签添加到资源 884
向容器实例添加标签 886
外部容器实例 887
使用情况报告 888
任务级成本和使用量 889
监控 891
监控 Amazon ECS 的最佳实践 891
监控工具 892
自动化工具 892
手动工具 893
使用 CloudWatch 监控 Amazon ECS 894
注意事项 894
推荐的指标 895
查看 Amazon ECS 指标 895
Amazon ECS CloudWatch 指标 897
AWS Fargate 使用情况指标 905
Amazon ECS 集群预留指标 906
Amazon ECS 集群利用率指标 907
Amazon ECS 服务利用率指标 909
使用 EventBridge 自动响应 Amazon ECS 错误 911
Amazon ECS Events 912
处理事件 929
使用 Container Insights 监控 Amazon ECS 容器 932
注意事项 933
为 Amazon ECS 配置 CloudWatch Container Insights 933
CloudWatch Container Insights 查看 Amazon ECS 生命周期事件所需的权限 934
使用容器运行状况检查确定任务运行状况 936
如何确定任务运行状况 937
运行状况检查和代理断开连接 938
查看容器运行状况 938
监控 Amazon ECS 容器实例运行状况 938

相关 主题 939
使用应用程序跟踪数据来识别 Amazon ECS 的优化机会 939
AWS Distro for OpenTelemetry 与 AWS X-Ray 集成所需的 IAM 权限 940
指定 AWS Distro for OpenTelemetry 附加，用于任务定义中的 AWS X-Ray 集成 941
使用应用程序指标关联 Amazon ECS 应用程序性能 943
将应用程序指标导出到 Amazon CloudWatch 943
将应用程序指标导出到 Amazon Managed Service for Prometheus 947
使用 AWS CloudTrail 记录 Amazon ECS API 调用 950
CloudTrail 中的 Amazon ECS 信息 951
了解 Amazon ECS 日志文件条目 952
使用元数据监控工作负载 953
容器元数据文件 954
任务元数据可用于 EC2 上的 Amazon ECS 任务 959
任务元数据可用于 Fargate 上的任务 999
容器自检 1020
使用运行时监控识别未经授权的行为 1023
运行时监控如何与 Amazon ECS 结合使用 1023
注意事项 1024
资源利用率 1025
Fargate 工作负载的运行时监控 1025
EC2 工作负载的运行时监控 1029
问题排查常见问题 1034
使用 ECS Exec 监控 Amazon ECS 容器 1038
注意事项 1038
先决条件 1040
架构 1040
使用 ECS Exec 1041
使用 ECS Exec 进行日志记录 1043
使用 IAM policy 限制对 ECS Exec 的访问权限 1047
Compute Optimizer 建议 1050
为 Fargate 生成的任务大小建议 1050
故障排除 1052
解决已停止任务错误 1054
已停止任务错误消息更新 1055
查看已停止任务错误 1057
已停止的任务错误代码 1058

验证任务连接 1076
查看 IAM 角色请求 1080
查看服务事件消息 1081
Amazon ECS 服务事件消息 1082
对 Amazon ECS 中的服务负载均衡器进行故障排除 1090
对 Amazon ECS 中的服务自动扩缩进行故障排除 1092
排查任务定义 CPU 或内存无效错误 1092
查看容器代理日志 1094
使用 Amazon ECS 日志收集器收集容器日志 1095
代理自检 1097
Amazon ECS 中的 Docker 诊断 1099
列出 Amazon ECS 中的 Docker 容器 1099
查看 Amazon ECS 中的 Docker 日志 1100
检查 Amazon ECS 中的 Docker 容器 1101
在 Amazon ECS 中配置 Docker 进程守护程序的详细输出 1102
对 Amazon ECS 中的 Docker API error (500): devmapper 进行故障排除 1104
排查 ECS Exec 问题 1105
使用 Exec Checker 进行验证 1105
调用 execute-command 时出错 1105
排查 Amazon ECS Anywhere 问题 1105
外部实例注册问题 1106
外部实例网络问题 1106
运行任务的问题 1107
AWS Fargate 限制配额 1107
在 Fargate 中对 RunTask API 进行节流 1108
在 Fargate 中调整速率配额 1108
处理节流问题 1108
同步节流 1108
Amazon ECS 中的异步节流 1109
监控节流 1109
使用 CloudWatch 监控节流 1110
API 失败原因 1111
安全性 1117
Identity and Access Management 1117
受众 1118
使用身份进行身份验证 1119

使用策略管理访问 1121
如何将 Amazon Elastic Container Service 与 IAM 结合使用 1123
基于身份的策略示例 1133
Amazon ECS 的 AWS 托管式策略 1144
使用服务相关角色 1172
适用于 Amazon ECS 的 IAM 角色 1175
Amazon ECS 控制台所需的权限 1222
Amazon ECS 服务自动扩缩所需的 IAM 权限 1228
在创建过程中，为资源添加标签 1229
故障排除 1233
IAM 最佳实践 1235
日志记录和监控 1237
合规性验证 1239
合规性和安全性最佳实践 1240
AWS Fargate FIPS-140 合规性 1241
AWS Fargate FIPS-140 注意事项 1242
在 Fargate 上使用 FIPS 1242
将 CloudTrail 用于 Fargate FIPS-140 审核 1242
基础设施安全性 1244
接口 VPC 终端节点 (AWS PrivateLink) 1244
任务和容器安全性最佳实践 1249
创建最小的映像或使用 distroless 映像 1249
扫描您的映像中是否存在漏洞 1250
移除映像的特殊权限 1251
创建一组精选映像 1251
扫描应用程序包和库中是否存在漏洞 1250
执行静态代码分析 1252
以非根用户身份运行容器 1252
使用只读的根文件系统 1252
使用 CPU 和内存限制配置任务（Amazon EC2） 1253
在 Amazon ECR 中使用不可变标签 1253
避免以特权身份运行容器（Amazon EC2） 1253
从容器中移除不必要的 Linux 功能 1253
使用客户自主管理型密钥（CMK）加密推送到 Amazon ECR 的映像 1254
教程 1255
使用 AWS CLI 创建 Fargate 启动类型的 Linux 任务 1256

先决条件 1257
步骤 1：创建集群 1258
第 2 步：注册 Linux 任务定义 1259
第 3 部：列出任务定义 1260
步骤 4：创建服务 1260
步骤 5：列出服务 1261
步骤 6：描述正在运行的服务 1261
步骤 7：测试 1264
步骤 8：清除 1267
使用 AWS CLI 创建 Fargate 启动类型的 Windows 任务 1268
先决条件 1268
步骤 1：创建集群 1269
步骤 2：注册 Windows 任务定义 1269
步骤 3：列出任务定义 1271
步骤 4：创建服务 1271
步骤 5：列出服务 1272
步骤 6：描述正在运行的服务 1272
步骤 7：清理 1274
使用 AWS CLI 创建 EC2 启动类型的任务 1275
先决条件 1275
步骤 1：创建集群 1275
步骤 2：使用 Amazon ECS AMI 启动实例 1276
第 3 步：列出容器实例 1276
第 4 步：描述容器实例 1277
第 5 步：注册任务定义 1280
第 6 部：列出任务定义 1281
第 7 步：运行任务 1282
第 8 步：列出任务 1283
第 9 步：描述正在运行的任务 1283
配置 Amazon ECS 以侦听 CloudWatch Events 事件 1284
先决条件：设置测试集群 1284
步骤 1：创建 Lambda 函数 1284
步骤 2：注册事件规则 1285
步骤 3：创建任务定义 1286
步骤 4：测试您的规则 1287
针对任务停止事件发送 Amazon Simple Notification Service 警报 1287

先决条件：设置测试集群 1287
先决条件：为 Amazon SNS 配置权限 1288
步骤 1：创建并订阅 Amazon SNS 主题 1288
步骤 2：注册事件规则 1289
步骤 3：测试您的规则 1290
连接多行或堆栈跟踪日志消息 1291
所需的 IAM 权限 1291
确定何时使用多行日志设置 1293
解析和连接选项 1294
在 Windows 容器上部署 Fluent Bit 1312
先决条件 1314
步骤 1：创建 IAM 访问角色 1315
步骤 2：创建 Amazon ECS Windows 容器实例 1316
步骤 3：配置 Fluent Bit 1317
步骤 4：注册 Windows Fluent Bit 任务定义，该定义将日志路由到 CloudWatch 1319
步骤 5：使用进程守护程序计划策略将 ecs-windows-fluent-bit 任务定义作为 Amazon ECS 服务运行 1321
步骤 6：注册生成日志的 Windows 任务定义 1322
步骤 7：运行 windows-app-task 任务定义 1323
步骤 8：验证 CloudWatch 上的日志 1324
步骤 9：清除 1325
将 gMSA 用于 EC2 Linux 容器 1326
注意事项 1326
先决条件 1327
设置 1328
CredSpec file 1334
将 gMSA 用于 Fargate 上的 Linux 容器 1335
注意事项 1335
先决条件 1336
设置 1336
CredSpec file 1339
通过 AWS CLI 将 Windows 容器与无域 gMSA 结合使用 1340
先决条件 1341
步骤 1：在 Active Directory 域服务（AD DS）上创建和配置 gMSA 账户 1342
步骤 2：将凭证上传到 Secrets Manager 1344
步骤 3：修改您的 CredSpec JSON 以包含无域 gMSA 信息 1345

步骤 4：将 CredSpec 上传到 Amazon S3 1346
步骤 5：（可选）创建 Amazon ECS 集群 1346
步骤 6：为容器实例创建 IAM 角色 1347
步骤 7：创建自定义任务执行角色 1347
步骤 8：为 Amazon ECS Exec 创建任务角色 1348
步骤 9：注册任务定义 1350
步骤 10：注册 Windows 容器实例 1351
步骤 11：验证容器实例 1352
步骤 12：运行 Windows 任务 1353
步骤 13：验证容器是否有 gMSA 凭证 1353
步骤 14：清除 1354
调试 1355
了解如何将 gMSA 用于 EC2 Windows 容器 1356
注意事项 1356
先决条件 1357
设置 1358
使用 Image Builder 构建自定义的 Amazon ECS 优化型AMI 1363
将映像 ARN 与基础设施即代码（IaC）结合使用 1364
将映像 ARN 与 AWS CloudFormation 结合使用 1367
将映像 ARN 与 Terraform 结合使用 1368
使用 AWS Deep Learning Containers 1368
Amazon ECS 上的 Elastic Inference Deep Learning Containers 1368
服务限额 1370
Amazon ECS 服务配额 1370
AWS Fargate 服务限额 1373
在 AWS Management Console 中管理您的服务配额 1374
处理服务配额和 API 节流限制 1375
Elastic Load Balancing 1376
弹性网络接口 1377
AWS Cloud Map 1379
Amazon ECS API 参考 1380
文档历史记录 1381

什么是 Amazon Elastic Container Service？

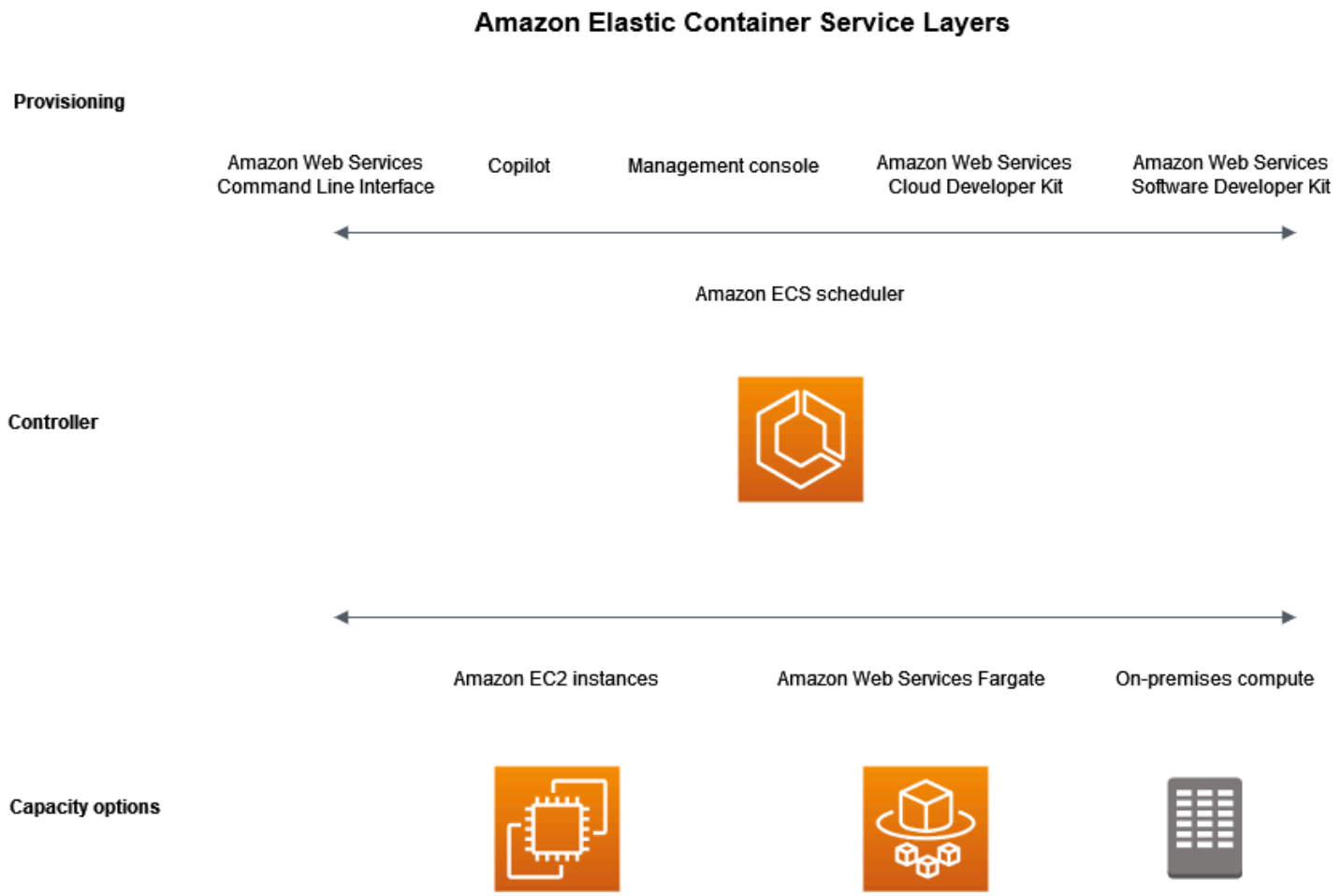
Amazon Elastic Container Service (Amazon ECS) 是一种完全托管式的容器编排服务，可以帮助您轻松部署、管理和扩展容器化应用程序。作为一种完全托管式服务，Amazon ECS 内置有 AWS 配置和操作最佳实践。它与 AWS 和第三方工具（例如 Amazon Elastic Container Registry 和 Docker）集成。这种集成使团队更容易专注于构建应用程序而不是环境。您可以在云端和本地跨 AWS 区域 运行和扩展容器工作负载，而无需以复杂的方式管理控制面板。

Amazon ECS 术语和组件

Amazon ECS 中有三个层级：

- 容量 - 容器运行所在的基础设施
- 控制器 - 部署和管理在容器上运行的应用程序
- 预调配 - 可用于与计划程序交互以部署和管理应用程序和容器的工具

下图显示了 Amazon ECS 层。



Amazon ECS 容量

Amazon ECS 容量是容器运行所在的基础设施。下文概述了容量选项：

- AWS Cloud 中的 Amazon EC2 实例

  您可以选择实例类型、实例数量并管理容量。

- AWS Cloud 中的无服务器（AWS Fargate (Fargate)）

  Fargate 是一款按实际使用量付费的无服务器计算引擎。使用 Fargate，您无需管理服务器、处理容量计划或隔离容器工作负载即可获得安全。

- 本地虚拟机（VM）或服务器

  Amazon ECS Anywhere 支持向 Amazon ECS 群集注册外部实例，如本地部署服务器或虚拟机（VM）。

容量可以位于以下任何 AWS 资源中：

- 可用区
- Local Zones
- Wavelength 区域
- AWS 区域
- AWS Outposts

Amazon ECS 控制器

Amazon ECS 计划程序是管理您的应用程序的软件。

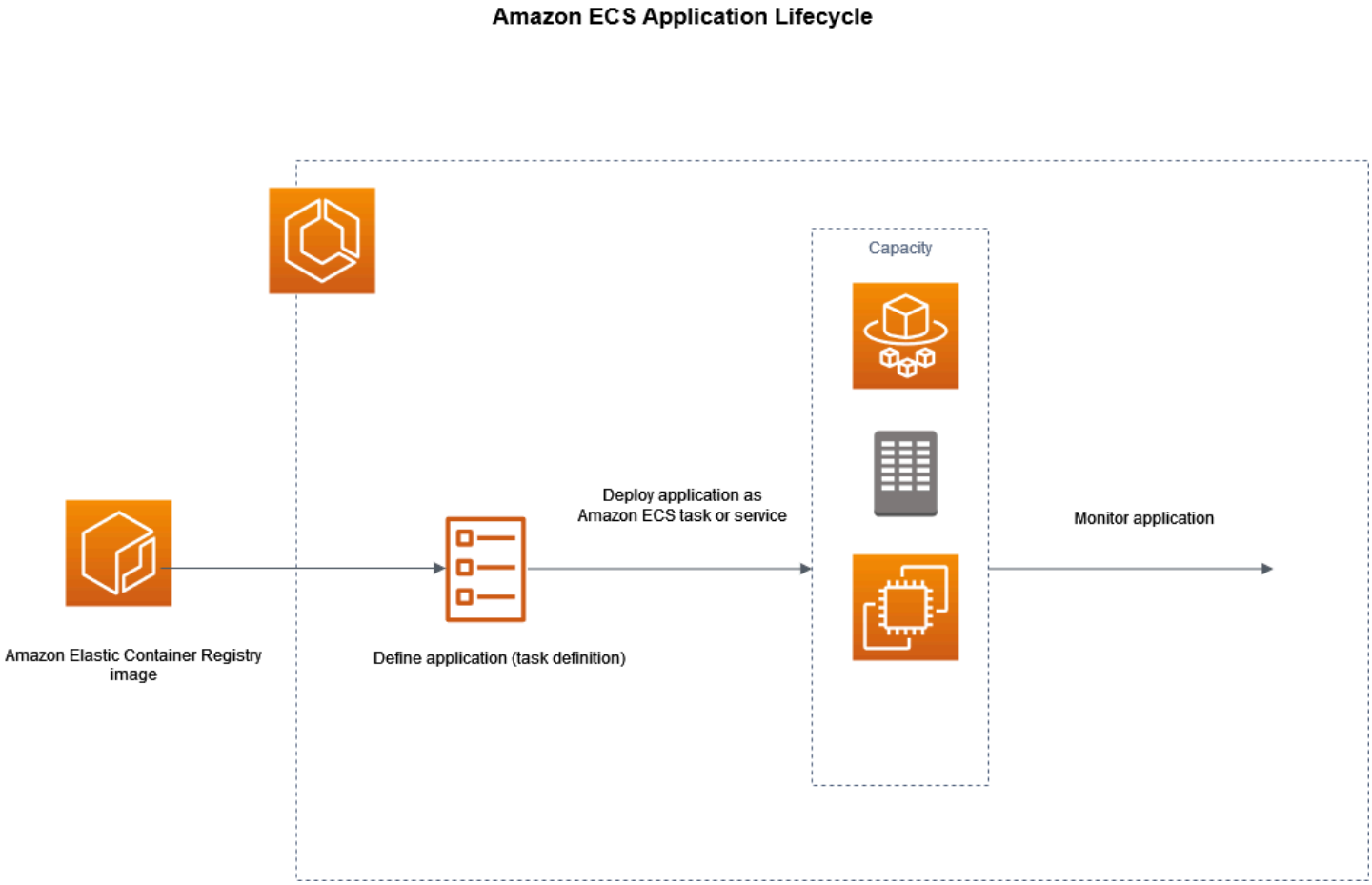
Amazon ECS 预调配

有多种预调配 Amazon ECS 的选项：

- AWS Management Console — 提供了一个可用来访问 Amazon ECS 资源的 Web 界面。
- AWS Command Line Interface（ ）AWS CLI— 为众多 AWS 服务（包括 Amazon ECS）提供命令。它在 Windows、Mac 和 Linux 上受支持。有关更多信息，请参阅 AWS Command Line Interface。
- AWS 开发工具包 — 提供特定于语言的 API 并关注许多连接详细信息。这些工具包括计算签名、处理请求重试和错误处理。有关更多信息，请参阅 AWS 软件开发工具包。
- Copilot – 为开发人员提供开源工具，以便在 Amazon ECS 上构建、发布和运行生产就绪型容器化应用程序。有关更多信息，请参阅 GitHub 网站上的 Copilot。
- AWS CDK — 提供开源软件开发框架，您可以通过它使用熟悉的编程语言对云应用程序资源进行建模和预置。AWS CDK 通过 AWS CloudFormation 以安全、可重复的方式预置资源。

应用程序生命周期

下图显示了应用程序生命周期及其如何与 Amazon ECS 组件配合使用。



您必须将应用程序设计为能够在容器上运行。容器是软件开发的标准化单位，它包含软件应用程序运行需要的所有内容。这包括相关代码、运行时、系统工具和系统库。从称为映像的只读模板中创建容器。通常通过 Dockerfile 进行构建镜像。Dockerfile 是一个纯文本文件，其中包含构建容器的说明。构建完成后，这些映像将存储在可从中下载它们的注册表，例如 Amazon ECR。

创建并存储映像后，创建 Amazon ECS 任务定义。任务定义是应用程序的蓝图。它是描述构成应用程序的参数和一个或多个容器的 JSON 格式的文本文件。例如，您可以使用它指定操作系统的映像和参数、要使用哪些容器、要为应用程序打开哪些端口以及任务中的容器应使用哪些数据卷。可用于任务定义的特定参数取决于您的特定应用程序需求。

定义任务定义后，将其部署为集群上的服务或任务。集群是在注册到集群的容量基础设施上运行的任务或服务的逻辑分组。

任务 是集群内的任务定义的实例化。您可以运行独立任务，也可以将任务作为服务的一部分运行。您可以使用 Amazon ECS 服务在 Amazon ECS 集群中同时运行和维护所需数量的任务。它的工作原理是，如果您的任何任务出于任何原因失败或停止，Amazon ECS 服务调度器将根据您的任务定义启动另一个实例。这样做是为了替换它，从而保持服务中所需的任务数量。

容器代理 在 Amazon ECS 群集内的每个实例上运行。代理向 Amazon ECS 发送有关容器当前正在运行的任务和资源使用率的信息。在接收来自 Amazon ECS 的请求时启动和停止任务。

部署任务或服务后，可以使用以下任一工具来监控部署和应用程序：

- CloudWatch
- 运行时监控

Amazon ECS 相关信息

下列相关资源在您使用此服务的过程中会有所帮助。

- AWS Fargate – Fargate 功能的概述。
- AWS 上的 Windows – AWS 上的 Windows 工作负载和服务概述。
- 来自 AWS 的 Linux – 来自 AWS 的基于 Linux 的现代操作系统组合。

开发人员教程

- AWS 计算博客 – 有关新功能的信息、对功能、代码示例和最佳实践的深入研究。

AWS re:Post

AWS re:Post – AWS 托管式问答（Q & A）服务，为您的技术问题提供众包、经专家审查的答案。

定价

- Amazon ECS 定价 – Amazon ECS 的定价信息。
- AWS Fargate 定价 – Fargate 的定价信息。

一般 AWS 资源

以下一般资源在您使用 AWS 的过程中会有所帮助。

- 课程和研讨会 – 指向基于角色的专业课程和自主进度动手实验室的链接，这些课程和实验室旨在帮助您增强 AWS 技能并获得实践经验。
- AWS 开发人员中心 – 浏览教程、下载工具并了解 AWS 开发人员活动。

- AWS 开发人员工具 – 指向开发人员工具、开发工具包、IDE 工具包和命令行工具的链接，这些资源用于开发和管理 AWS 应用程序。
- 入门资源中心 – 了解如何设置 AWS 账户、加入 AWS 社区和启动您的第一个应用程序。
- 动手实践教程 – 按照分步教程在 AWS 上启动您的第一个应用程序。
- AWS 白皮书 – 指向 AWS 技术白皮书的完整列表的链接，这些资料涵盖了架构、安全性、经济性等主题，由 AWS 解决方案架构师或其他技术专家编写。
- AWS Support 中心 – 用于创建和管理 AWS Support 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及AWS Trusted Advisor。
- AWS Support – 提供有关 AWS Support 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关 AWS 账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 – 有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

了解如何创建和使用 Amazon ECS 资源

以下指南介绍了可用于访问 Amazon ECS 的工具，以及运行容器的简要过程。Docker 基本知识将引导您完成创建 Docker 容器映像并将其上传到 Amazon ECR 私有存储库的基本步骤。入门指南会您使用 AWS Copilot命令行界面和 AWS Management Console 完成在 Amazon ECS 和 AWS Fargate 上运行容器的常见任务。

内容

- 设置以使用 Amazon ECS
- 创建容器镜像以在 Amazon ECS 上使用
- 了解如何创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 了解如何创建 Fargate 启动类型的 Amazon ECS Windows 任务
- 了解如何创建 EC2 启动类型的 Amazon ECS Windows 任务

设置以使用 Amazon ECS

如果您已经注册了Amazon Web Services (AWS)，并且一直在使用Amazon Elastic Compute Cloud (Amazon EC2)，那么您就接近能够使用 Amazon ECS 了。这两个服务的设置过程相似。以下指南为启动您的第一个 Amazon ECS 集群做好准备。

要开始设置 Amazon ECS，请完成以下任务。

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 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，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册 AWS 账户 后，请保护好您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关如何使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认的 IAM Identity Center 目录 配置用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

创建 Virtual Private Cloud

您可以使用 Amazon Virtual Private Cloud（Amazon VPC）将AWS资源启动到您定义的虚拟网络中。强烈建议您在 VPC 中启动您的容器实例。

如果您有默认 VPC，则可以跳过此部分并进入下一个任务，即 创建安全组。要确定您是否具有默认 VPC，请参阅《Amazon EC2 用户指南》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。

有关如何创建 VPC 的信息，请参阅《Amazon VPC 用户指南》中的仅创建 VPC 并使用下表确定要选择的选项。

选项	值
要创建的资源	仅限 VPC
名称	可以选择为您的 VPC 提供名称。
IPv4 CIDR 块	IPv4 CIDR 手动输入 CIDR 块大小必须在 /16 和 /28 之间。
IPv6 CIDR 块	无 IPv6 CIDR 块
租赁	默认

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

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。

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》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”。如果您通过互联网服务提供商 (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。

有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的创建安全组并使用下表确定要选择的选项。

选项	值
区域	您在其中创建密钥对的相同区域。
名称	一个您容易记住的名称，例如 ecs-instances-default-cluster。
VPC	默认 VPC（标有“（默认）”）。 **Note** 如果您的账户支持 Amazon EC2 Classic，请选择您在

选项	值
	上一个任务中创建的 VPC。

有关要为您的应用场景添加的出站规则的信息，请参阅《Amazon EC2 用户指南》中的针对不同应用场景的安全组规则。

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。

向您的安全组添加以下三条入站规则。有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的向安全组添加规则。

选项	值
HTTP 规则	类型：HTTP 来源：Anywhere（任何位置）（`0.0.0.0/0` ） 此选项将自动添加 0.0.0.0/0 IPv4 CIDR 块作为源。这在测试环境中可以接受一小段时间，但是在生产环境中并不安全。在生产中，请仅授权特定 IP 地址或地址范围访问您的实例。
HTTPS 规则	类型：HTTPS 来源：Anywhere（任何位置）（`0.0.0.0/0` ） 这在测试环境中可以接受一小段时间，但是在生产环境中并

选项	值
	不安全。在生产中，请仅授权特定 IP 地址或地址范围访问您的实例。
SSH 规则	类型：SSH 来源：Custom（自定义），使用 CIDR 表示法指定计算机的公有 IP 地址或网络。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）。

创建凭证以连接到 EC2 实例

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》中的区域和可用区。

创建密钥对

- 使用 Amazon EC2 控制台创建密钥对。有关创建密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的创建密钥对。

有关如何连接到实例的信息，请参阅《Amazon EC2 用户指南》中的连接到 Linux 实例。

安装 AWS CLI

AWS Management Console 可以用来手动管理 Amazon ECS 的所有操作。但是，您可以在本地桌面或开发人员工具包上安装 AWS CLI，以便构建可在 Amazon ECS 中自动执行常见管理任务的脚本。

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。

创建容器镜像以在 Amazon ECS 上使用

Amazon ECS 在任务定义中使用 Docker 镜像来启动容器。Docker 技术为您提供了在容器中构建、运行、测试和部署分布式应用程序的工具。

此处概述的步骤的目的是指导您创建第一个 Docker 镜像并将该映像推送到 Amazon ECR（容器注册表），以便在 Amazon ECS 任务定义中使用。此演练假定您已基本了解 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。

先决条件

在您开始之前，确保您满足以下先决条件。

- 确保您已完成 Amazon ECR 设置步骤。有关更多信息，请参阅《Amazon Elastic Container Registry 用户指南》中的设置 Amazon ECR。
- 您的用户具有访问和使用 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。
- 您已安装 Docker。有关 Amazon Linux 2 的 Docker 安装步骤，请参阅 在 AL2023 上安装 Docker。对于所有其他操作系统，请参阅 Docker 桌面概述中的 Docker 文档。

- 您已经安装并配置 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS Command Line Interface。

如果您没有或不需要本地开发环境，并且更喜欢使用 Amazon EC2 实例来使用 Docker，我们将提供以下步骤来使用 Amazon Linux 2 启动 Amazon EC2 实例并安装 Docker Engine 和 Docker CLI。

在 AL2023 上安装 Docker

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 MacOS 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。

您无需本地开发系统即可使用 Docker。如果您已在使用 Amazon EC2，则可启动 Amazon Linux 2023 实例并安装 Docker 以开始使用。

如果您已安装 Docker，请跳到创建 Docker 映像。

使用 Amazon Linux 2023 AMI 在 Amazon EC2 实例上安装 Docker

1. 使用最新版 Amazon Linux 2023 AMI 启动实例。有关更多信息，请参阅《Amazon EC2 用户指南》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》中的连接到 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 yum 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 进程守护程序的权限。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 镜像启动群集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 镜像，并在本地系统或 Amazon EC2 实例上测试此映像，然后将此映像推送至 Amazon ECR 容器注册表，以便能够在 Amazon ECS 任务定义中使用它。

创建简单 Web 应用程序的 Docker 镜像

1. 创建名为 Dockerfile 的文件。Dockerfile 是一个清单文件，描述了用于 Docker 镜像的基本镜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 public.ecr.aws/amazonlinux/amazonlinux:latest

   # Install dependencies
   RUN yum update -y && \
    yum install -y httpd

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

   # Configure apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
   ```

```
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

此 Dockerfile 使用 Amazon ECR Public 上托管的 Amazon Linux 2 公有映像。RUN 指令更新包缓存，安装一些适用于 Web 服务器的软件包，然后将“Hello World!” 内容写入 Web 服务器的文档根目录。EXPOSE 指令指的是容器的端口 80 为正在侦听的端口，CMD 指令启动 Web 服务器。

3. 从您的 Dockerfile 生成 Docker 镜像。

> **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. 列出容器映像。

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

输出：

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. 运行新构建的镜像。-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 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。

将映像推送到 Amazon Elastic Container Registry

Amazon ECR 是一项托管 AWS Docker 注册表服务。您可以使用 Docker CLI 在 Amazon ECR 存储库中推送、拉取和托管映像。有关 Amazon ECR 产品详细信息、特色客户案例研究和常见问题解答，请参阅 Amazon Elastic Container Registry 产品详细信息页面。

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`。

   将 `region` 替换为您的 AWS 区域，例如 `us-east-1`。

   ```
   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-password 命令。指定要对其进行身份验证的注册表 URI。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的注册表身份验证。

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

输出：

```
Login Succeeded
```

> **Important**
>
> 如果收到错误，请安装或更新到最新版本的 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS Command Line Interface。

4. 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
```

清理

要继续创建 Amazon ECS 任务定义并使用容器镜像启动任务，请跳至 后续步骤。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

您的任务定义需要任务执行角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

在创建容器映像并将其推送到 Amazon ECR 后，您可以在任务定义中使用该映像。有关更多信息，请参阅以下章节之一：

- the section called “了解如何创建 Fargate 启动类型的 Linux 任务”
- the section called “了解如何创建 Fargate 启动类型的 Windows 任务”
- 使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Linux 任务

了解如何创建 Fargate 启动类型的 Amazon ECS Linux 任务

Amazon Elastic Container Service (Amazon ECS) 是一项高度可扩展的快速容器管理服务，可让您轻松运行、停止和管理容器。您可以通过在 AWS Fargate 上启动服务或任务，将容器托管在由 Amazon ECS 管理的无服务器基础设施上。有关 Fargate 的更多信息，请参阅 适用于 Amazon ECS 的 AWS Fargate。

通过在区域（这些区域中的 Amazon ECS 支持 AWS Fargate）中将 Fargate 启动类型用于您的任务，开始在 AWS Fargate 上使用 Amazon ECS。

要在 AWS Fargate 上开始使用 Amazon ECS，请完成以下步骤。

先决条件

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤，并且您的 AWS 用户具有 `AdministratorAccess` IAM policy 示例中指定的权限。

控制台会尝试自动创建任务执行 IAM 角色，这是 Fargate 任务需要的。要确保控制台能够创建该 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 设置以使用 Amazon ECS。

- 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。

> **Important**
>
> 使用任务定义创建服务时所选择的安全组，必须为入站流量开放端口 80。将以下入站规则添加到安全组。有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的向安全组添加规则。
>
> - 类型：HTTP
> - 协议：TCP
> - 端口范围：80
> - 来源：Anywhere（任何位置）（`0.0.0.0/0`）

第 1 步：创建集群

创建使用默认 VPC 的集群。

在开始之前，分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在 Cluster configuration（集群配置）下，为 Cluster name（集群名称），输入唯一名称。

   该名称最多可以包含 255 个字母（大小写字母）、数字和连字符。
6. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。
7. （可选）为了帮助识别您的集群，请展开 Tags（标签），然后配置您的标签。

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：

- 对于 Key（键），输入键名称。
- 对于值，输入键值。

[删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。

8. 选择创建。

第 2 步：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. 在导航窗格中，选择 Task Definitions。
2. 选择 Create new Task Definition（创建新的任务定义）、Create new revision with JSON（使用 JSON 创建新的修订）。
3. 将以下示例任务定义复制并粘贴到框中，然后选择 Save (保存)。

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "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"
}
```

4. 选择创建。

第 3 步：创建服务

创建使用该任务定义的服务。

1. 在导航窗格中，选择 Clusters（集群），然后选择您在 第 1 步：创建集群 中创建的集群。
2. 在 Services（服务）选项卡上，选择 Create（创建）。
3. 在 Deployment configuration（部署配置）下，指定应用程序的部署方式。
   a. 在 Task Definitions（任务定义）中，选择您在 第 2 步：创建任务定义 中创建的任务定义。
   b. 对于 Service name（服务名称），为您的服务输入一个名称。
   c. 对于 Desired tasks（所需任务），输入 1。
4. 在联网下，您可以为您的任务创建新安全组或者选择现有安全组。确保您使用的安全组具有 先决条件 下列出的入站规则。
5. 选择创建。

步骤 4：查看您的服务

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 选择您在其中运行服务的集群。
4. 在服务选项卡中的服务名称下，选择您在 第 3 步：创建服务 中创建的服务。

5. 选择任务选项卡，然后在服务中选择任务。
6. 在“任务”页面的配置部分，在公有 IP 的下方选择打开地址。

第 5 步：清理

完成使用 Amazon ECS 集群后，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 EC2 实例、Elastic Load Balancing和自动扩缩组）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。

1. 在导航窗格中，选择集群。
2. 在集群页面上，选择您为本教程而创建的集群。
3. 选择服务选项卡。
4. 选择服务，然后选择删除。
5. 在确认提示符处，输入 delete，然后选择 Delete（删除）。您也可以使用 `Force delete` 选项，让 Amazon ECS 在将其删除之前代表您缩减服务。

   等待直至系统删除服务。
6. 选择 Delete Cluster (删除集群)。在确认提示符中，输入 delete ***`cluster-name`***，然后选择 Delete（删除）。删除该集群将清除使用该集群创建的关联资源，包括自动扩缩组、VPC 或负载均衡器。

了解如何创建 Fargate 启动类型的 Amazon ECS Windows 任务

通过在区域（这些区域中的 Amazon ECS 支持 AWS Fargate）中将 Fargate 启动类型用于您的任务，开始在 AWS Fargate 上使用 Amazon ECS。

要在 AWS Fargate 上开始使用 Amazon ECS，请完成以下步骤。

先决条件

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤，并且您的 AWS 用户具有 `AdministratorAccess` IAM policy 示例中指定的权限。

控制台会尝试自动创建任务执行 IAM 角色，这是 Fargate 任务需要的。要确保控制台能够创建该 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 设置以使用 Amazon ECS。
- 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。

Important

使用任务定义创建服务时所选择的安全组，必须为入站流量开放端口 80。将以下入站规则添加到安全组。有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的向安全组添加规则。

- 类型：HTTP
- 协议：TCP
- 端口范围：80
- 来源：Anywhere（任何位置）（`0.0.0.0/0`）

步骤 1：创建集群

您可以创建名为 windows 并且使用默认 VPC 的新集群。

使用 AWS Management Console创建集群

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在 Cluster configuration（集群配置）下方的 Cluster name（集群名称）中，输入 windows。
6. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。
7. （可选）为了帮助识别您的集群，请展开 Tags（标签），然后配置您的标签。

[添加标签] 选择 Add tag（添加标签），然后执行以下操作：

- 对于 Key（键），输入键名称。
- 对于值，输入键值。

[删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。

8. 选择创建。

步骤 2：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 `mcr.microsoft.com/windows/servercore/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 Management Console注册示例任务定义

1. 在导航窗格中，选择 Task definitions（任务定义）。
2. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
3. 将以下示例任务定义复制并粘贴到框中，然后选择 Save (保存)。

```
{
    "containerDefinitions": [
        {
            "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"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
```

```
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

4. 验证您的信息并选择 Create。

步骤 3：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. 在导航窗格中，选择 Clusters（集群），然后选择您在 步骤 1：创建集群 中创建的集群。
2. 在 Services（服务）选项卡上，选择 Create（创建）。
3. 在 Deployment configuration（部署配置）下，指定应用程序的部署方式。
   a. 在 Task Definitions（任务定义）中，选择您在 步骤 2：注册 Windows 任务定义 中创建的任务定义。
   b. 对于 Service name（服务名称），为您的服务输入一个名称。
   c. 对于 Desired tasks（所需任务），输入 1。
4. 在联网下，您可以为创建安全组或者选择现有安全组。确保您使用的安全组具有 先决条件 下列出的入站规则。

5. 选择创建。

步骤 4：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/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 选择您在其中运行服务的集群。
4. 在服务选项卡中的服务名称下，选择您在 步骤 3：使用您的任务定义创建服务 中创建的服务。
5. 选择任务选项卡，然后在服务中选择任务。
6. 在“任务”页面的配置部分，在公有 IP 的下方选择打开地址。

第 5 步：清除

完成使用 Amazon ECS 集群后，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 EC2 实例、Elastic Load Balancing和自动扩缩组）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。

1. 在导航窗格中，选择集群。
2. 在集群页面上，选择您为本教程而创建的集群。
3. 选择服务选项卡。
4. 选择服务，然后选择删除。
5. 在确认提示符处，输入 delete，然后选择 Delete（删除）。

等待直至系统删除服务。

6. 选择 Delete Cluster (删除集群)。在确认提示符中，输入 delete ***cluster-name***，然后选择 Delete（删除）。删除该集群将清除使用该集群创建的关联资源，包括自动扩缩组、VPC 或负载均衡器。

了解如何创建 EC2 启动类型的 Amazon ECS Windows 任务

通过在控制台中注册任务定义、创建集群和创建服务，开始使用利用 EC2 启动类型的 Amazon ECS。

完成以下步骤，开始使用 EC2 启动类型的 Amazon ECS。

先决条件

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤，并且您的 AWS 用户具有 `AdministratorAccess` IAM policy 示例中指定的权限。

控制台会尝试自动创建任务执行 IAM 角色，这是 Fargate 任务需要的。要确保控制台能够创建该 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 设置以使用 Amazon ECS。
- 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。

> **Important**
>
> 使用任务定义创建服务时所选择的安全组，必须为入站流量开放端口 80。将以下入站规则添加到安全组。有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的向安全组添加规则。
>
> - 类型：HTTP
> - 协议：TCP
> - 端口范围：80
> - 来源：Anywhere（任何位置）（`0.0.0.0/0`）

步骤 1：创建集群

Amazon ECS 集群是任务、服务和容器实例的逻辑分组。

以下步骤将指导您创建一个集群，该集群具有一个注册的 Amazon EC2 实例，这使我们能够在集群上运行任务。如果未提及特定字段，请保留默认的控制台值。

要创建新集群（Amazon ECS 控制台）

在开始之前，分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在 Cluster configuration（集群配置）下，为 Cluster name（集群名称），输入唯一名称。

   该名称最多可以包含 255 个字母（大小写字母）、数字和连字符。
6. （可选）要更改任务和服务启动所在的 VPC 和子网，在 Networking（联网）下，执行以下任一操作：

   - 要删除子网，请在 Subnets（子网）下，为您要删除的每个子网选择 X。
   - 要更改为默认 VPC 以外的 VPC，在 VPC 下，选择现有的 VPC，然后在 Subnets（子网）中，选择每个子网。
7. 要向集群添加 Amazon EC2 实例，请展开基础设施，然后选择 Amazon EC2 实例。接下来，配置充当容量提供程序的 自动扩缩组：

   a. 要使用现有 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择组。

   b. 要创建 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择 Create new group（创建新组），然后提供有关组的以下详细信息：

      - 对于Operating system/Architecture（操作系统/架构），为 自动扩缩组实例选择经 Amazon ECS 优化的 AMI。
      - 对于 EC2 instance type（EC2 实例类型），选择工作负载的实例类型。有关不同的实例类型更多信息，请参阅 Amazon EC2 实例。

        如果 自动扩缩组使用相同或相似的实例类型，则托管扩展效果最佳。

- 对于 SSH key pair（SSH 密钥对），连接到实例时，选择可证明您身份的密钥对。
- 对于 Capacity（容量），输入 自动扩缩组中启动的实例数的最小值和最大值。您的 AWS 资源中有 Amazon EC2 实例时会产生成本。有关更多信息，请参阅 Amazon EC2 定价。

8. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。
9. （可选）要管理集群标签，请展开 Tags（标签），然后执行以下操作之一：

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：

   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。

   [删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。
10. 选择创建。

第 2 步：注册任务定义

使用 AWS Management Console注册示例任务定义

1. 在导航窗格中，选择 Task Definitions。
2. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
3. 将以下示例任务定义复制并粘贴到框中，然后选择保存。

```
{
    "containerDefinitions": [
        {
            "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"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
```

```
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
}
```

4. 验证您的信息并选择 Create。

步骤 3：创建服务

Amazon ECS 服务可帮助您在 Amazon ECS 集群中同时运行和维护指定数量的任务定义实例。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以便在服务中维护所需数量的任务。有关服务的更多信息，请参阅 Amazon ECS 服务。

创建服务

1. 在导航窗格中，选择集群。
2. 选择您在 步骤 1：创建集群 中创建的集群。
3. 在 Services (服务) 选项卡上，选择 Create (创建)。
4. 在 Environment（环境）部分中，执行以下操作：
   a. 针对 Compute options（计算选项），选择 Launch type（启动类型）。
   b. 对于 Launch type (启动类型)，选择 EC2
5. 在 Deployment configuration（部署配置）部分中，执行以下操作。

a. 在 Family（系列）中，选择您在 第 2 步：注册任务定义 中创建的任务定义。
b. 对于 Service name（服务名称），为您的服务输入一个名称。
c. 对于 Desired tasks（所需任务），输入 1。

6. 查看选项并选择创建。
7. 选择 View service (查看服务) 以查看您的服务。

步骤 4：查看您的服务

此服务是一个基于 Web 的应用程序，因此您可以使用 Web 浏览器查看其容器。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 选择您在其中运行服务的集群。
4. 在服务选项卡中的服务名称下，选择您在 步骤 3：创建服务 中创建的服务。
5. 选择任务选项卡，然后在服务中选择任务。
6. 在“任务”页面的配置部分，在公有 IP 的下方选择打开地址。以下屏幕截图为预期的输出。

第 5 步：清除

完成使用 Amazon ECS 集群后，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 EC2 实例、Elastic Load Balancing和自动扩缩组）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。

1. 在导航窗格中，选择集群。
2. 在集群页面上，请选择您为本教程创建的集群。
3. 选择服务选项卡。
4. 选择服务，然后选择删除。
5. 在确认提示符处，输入 delete，然后选择 Delete（删除）。

   等待直至系统删除服务。
6. 选择 Delete Cluster (删除集群)。在确认提示符中，输入 delete ***cluster-name***，然后选择 Delete（删除）。删除该集群将清除使用该集群创建的关联资源，包括自动扩缩组、VPC 或负载均衡器。

Amazon ECS 开发人员工具概述

无论是大型企业还是初创企业，无论专业知识水平如何，Amazon ECS 都提供多种工具，可帮助您快速启动和运行容器。您可以通过以下方式使用 Amazon ECS：

- 使用 AWS Management Console 了解、开发、管理和可视化容器应用程序和服务。
- 通过编程或使用 AWS Command Line Interface、AWS 开发工具包 或 ECS API，用自动化部署对 Amazon ECS 资源执行特定操作。
- 使用 AWS CloudFormation 自动部署定义和管理环境中的所有 AWS 资源。
- 使用完整的 AWS Copilot CLI 端到端开发人员工作流来创建、发布和操作符合基础架构 AWS 最佳实践的容器应用程序。
- 使用您首选的编程语言，将基础结构或体系结构定义为带有 AWS CDK 的代码。
- 通过使用容器的 AWS App2Container 集成可移植性和工具生态系统，将托管在本地或 Amazon EC2 实例上或两者上的应用程序容器化。
- 将应用程序部署到 Amazon ECS，或使用 Amazon ECS CLI 在 Amazon ECS 中运行的使用 Docker Compose 文件格式的容器测试本地容器。
- 从 与 Amazon ECS 集成的 Docker Desktop 中启动容器使用 Docker 桌面中的 Amazon ECS。

AWS Management Console

AWS Management Console 是用于管理 Amazon ECS 资源的基于浏览器的界面。该控制台提供了服务的直观概述，无需使用其他工具，您可以轻松地探索 Amazon ECS 的特性和功能。提供了许多相关教程和演练，可以指导您使用控制台。

有关指导您使用控制台的教程，请参阅 了解如何创建和使用 Amazon ECS 资源。

在开始时，许多客户更喜欢使用控制台，因为它提供了关于他们采取的行动是否成功的即时视觉反馈。熟悉 AWS Management Console 的 AWS 客户可以轻松管理相关资源，如负载平衡器和 Amazon EC2 实例。

从 AWS Management Console 开始

AWS Command Line Interface

AWS Command Line Interface (AWS CLI) 是用于管理 AWS 服务的统一工具。单独使用这个工具，您可以控制多个 AWS 服务，并通过脚本自动执行这些服务。AWS CLI 中的 Amazon ECS 命令是 Amazon ECS API 的反映。

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

AWS CLI 适用于喜欢并习惯于使用命令行工具编写脚本并与之交互的客户，他们确切知道希望在其 Amazon ECS资源 上执行哪些操作。AWS CLI 对于希望熟悉Amazon ECS API的客户也很有帮助。客户可以使用 AWS CLI 直接从命令行界面对 Amazon ECS 资源执行多项操作，包括创建、读取、更新和删除操作。

如果您熟悉或希望熟悉 Amazon ECS API 和相应的 CLI 命令，并希望编写自动脚本并在 Amazon ECS 资源上执行特定操作，使用 AWS CLI。

AWS CloudFormation

AWS CloudFormation 和 Terraform 为 Amazon ECS 提供了强大的方法，让您可以将基础设施定义为代码。您可以轻松跟踪您的模板版本或随时运行 AWS CloudFormation 堆栈，并在需要时回滚到以前的版本。您可以以相同的自动化方式执行基础架构和应用程序部署。这种灵活性和自动化使得 AWS CloudFormation 和 Terraform 两种常用格式，用于从连续交付管道将工作负载部署到 Amazon ECS。

有关 AWS CloudFormation 的更多信息，请参阅 使用 AWS CloudFormation 创建 Amazon ECS 资源。

使用 AWS CloudFormation 或 Terraform，如果您希望在 Amazon ECS 上自动执行基础设施部署和应用程序，并明确定义和管理所有AWS环境中的资源。

AWS Copilot CLI

AWS Copilot CLI（命令行界面）是一个全面的工具，使客户能够直接从源代码在 Amazon ECS 上部署和操作包装在容器和环境中的应用程序。使用AWS Copilot 时，您可以执行这些操作，而无需了解AWS和了解 Amazon ECS 元素，如应用程序负载平衡器、公共子网、任务、服务和集群 AWS。Copilot AWS 代表您从固定的服务模式（如负载平衡的web服务或后端服务）创建资源，为容

器化应用程序提供即时生产环境。您可以通过 AWS CodePipeline 管道跨多个环境、账户或区域，所有这些都可以在 CLI 内进行管理。通过使用 AWS Copilot，您还可以执行操作员任务，例如查看日志和服务运行状况。AWSCopilot 是一款多功能合一工具，可帮助您更轻松地管理云资源，让您可以专注于开发和管理应用程序。

有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。

使用 AWS Copilot complete-to-end 开发人员工作流创建、发布AWS和操作符合基础架构最佳实践的容器应用程序。

AWS CDK

AWS Cloud Development Kit (AWS CDK) 是一个开源软件开发框架，使您可以使用熟悉的编程语言对云应用程序资源进行建模和预置。通过 AWS CloudFormation，AWS CDK 能够以安全、可重复的方式预置资源。使用 CDK，客户可以使用与构建应用程序时使用的语言相同的语言，使用更少的代码行生成环境。Amazon ECS 在 CDK 中提供了一个名为 `ecs-patterns` 的模块，创建了通用的体系结构。一个可用的模式是 `ApplicationLoadBalancedFargateService()`。此模式创建集群、任务定义和其他资源，以便在 AWS Fargate 上运行负载平衡的 Amazon ECS 服务。

有关更多信息，请参阅 使用 AWS CDK 创建 Amazon ECS 资源。

如果要将基础结构或体系结构定义为首选编程语言中的代码，请使用 AWS CDK。例如，您可以使用与编写您的应用程序相同的语言。

AWS App2Container

有时企业客户可能已经拥有托管在本地或 EC2 实例上（或两者）的应用程序。他们对 Amazon ECS 上容器的可移植性和工具生态系统感兴趣，需要首先进行容器化。AWSApp2Container 使您能够做到这一点。App2Container (A2C) 是一个命令行工具，用于将 .NET 和 Java 应用程序现代化为容器化应用程序。A2C 分析并构建在虚拟机、本地部署或云中运行的所有应用程序的清单。选择要进行容器化的应用程序后，A2C将应用程序工件和标识的依赖项打包到容器映像中。然后，它会配置网络端口并生成 Amazon ECS 任务。最后，它会创建一个 CloudFormation 模板，您可以根据需要部署或修改该模板。

有关更多信息，请参阅 AWS App2Containe 入门。

如果您的应用程序托管在本地部署或 Amazon EC2 实例上（或两者），请使用 App2Container。

Amazon ECS CLI

通过 Amazon ECS CLI，您可以在 Amazon ECS 和 AWS Fargate 上使用 Docker Compose 文件格式运行应用程序。您可以快速配置资源，使用 Amazon ECR 推送并拉取映像，并监控 Amazon ECS 上正在运行的应用程序或 AWS Fargate。您还可以在 CLI 中测试本地运行的容器以及云中的容器。

有关更多信息，请参阅 开始使用 Amazon ECS 命令行界面。

如果您有 Compose 应用程序并希望将其部署到 Amazon ECS，或者使用云中 Amazon ECS中运行的容器测试本地容器，请使用 ECS CLI。

与 Amazon ECS 集成的 Docker Desktop

AWS 和 Docker 合作打造简化的开发人员体验，使您能够直接使用 Docker 工具在 Amazon ECS 上部署和管理容器。现在，您可以使用 Docker Desktop 和 Docker Compose 本地构建和测试容器，然后将它们部署到 Fargate 上的 Amazon ECS。要开始使用Amazon ECS和 Docker 集成，请下载 Docker Desktop 并选择注册 Docker ID。有关更多信息，请参阅 Docker Desktop 和 Docker ID 注册。

容器的初学者通常会使用 Docker CLI 和 Docker Compose等 Docker 工具开始学习容器。这样，使用 Docker Compose CLI 插件进行 Amazon ECS 成为在本地测试后 AWS 上运行容器的下一步。Docker 提供了有关在 Amazon ECS上部署容器的演练。有关更多信息，请参阅 Docker Compose CLI – Amazon ECS。

您可以利用其他 Amazon ECS 功能，例如服务发现、负载平衡和其他 AWS 资源与 Docker Desktop 配合使用的应用程序。

您还可以直接从 GitHub 下载 Amazon ECS 的 Docker Compose CLI 插件。有关更多信息，请参阅 GitHub 上针对 Amazon ECS 的 Docker Compose CLI插件。

AWS 开发工具包

您还可以使用 AWS SDK 用于通过各种编程语言管理 Amazon ECS 资源和操作。软件开发工具包提供了帮助处理任务的模块，包括以下列表中的任务。

- 对服务请求进行加密签名
- 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 Amazon Web Services 的工具。

Summary

有很多选项可供选择，您可以选择最适合您的选项。请考虑以下选项。

- 如果您是以视觉为导向的，则可以使用 AWS Management Console 创建并操作容器。
- 如果您更喜欢 CLI，请考虑使用 AWS Copilot 或 AWS CLI。如果您更喜欢 Docker 生态系统，您可以利用 Docker CLI 中的ECS功能部署到 AWS。部署这些资源后，您可以继续通过 CLI 或控制台进行直观管理。
- 如果您是一名开发人员，您可以使用 AWS CDK 通过与应用程序相同的语言定义基础结构。您可以使用 CDK 和 AWS Copilot导出到 CloudFormation 模板，您可以更改粒度设置、添加其他AWS资源，并通过脚本或 CI/CD 管道（如 CodePipeline）AWS自动化部署。

AWS CLI、软件开发工具包或 ECS API 是在 ECS 资源上自动执行操作的有用工具，非常适合部署。要使用 AWS CloudFormation 部署应用程序，可以使用多种编程语言或简单的文本文件来建模和提供应用程序所需的所有资源。然后，您可以以自动安全的方式跨多个区域和账户部署您的应用程序。例如，您可以将 ECS 群集、服务、任务定义或容量提供程序定义为文件中的代码，并通过 AWS CLI CloudFormation 命令进行部署。

要执行操作任务，您可以使用 AWS CLI、软件开发工具包或 ECS API。`describe-tasks` 或 `list-services` 等命令显示最新的元数据或所有资源的列表。与部署类似，客户可以编写一个自动化，其中包含 `update-service`，以便在检测到意外停止的资源时提供纠正措施。您也可以使用 AWS Copilot 操作您的服务。`copilot svc logs` 或 `copilot app show` 等命令提供有关每个微服务或整个应用程序的详细信息。

客户可以使用本文档中提到的任何可用刀具，并以各种组合方式使用。ECS工具提供了多种途径，可以从某些工具升级到使用其他适合您不断变化需求的工具。例如，您可以根据需要选择对资源进行更精细的控制或更多的自动化。ECS还提供多种工具，满足各种需求和专业水平。

使用 AWS Copilot 命令行界面创建 Amazon ECS 资源

AWS Copilot 命令行界面（CLI）命令简化了从本地开发环境在Amazon ECS上构建、发布和操作生产就绪的容器化应用程序。AWS Copilot CLI与支持现代应用程序最佳实践的开发人员工作流保持一致：从将基础结构用作代码到创建代表用户配置的 CI/CD 管道。将 AWS Copilot CLI 用作日常开发和测试周期的一部分，替代 AWS Management Console。

AWS Copilot 目前支持 Linux、macOS 和 Windows 系统。有关最新版本 AWS Copilot CLI 的更多信息，请参阅发行。

> **Note**
>
> AWS Copilot CLI的源代码可在 GitHub 上获得。我们建议您提交问题并请求更改您希望包含的内容。但是，Amazon Web Service 目前不支持运行 AWS Copilot 代码的修改副本。通过在 Gitter 或 GitHub 上与我们联系，向 AWS Copilot报告问题，您可以在那里打开问题、提供反馈和报告错误。

有关安装 AWS Copilot CLI的信息，请参阅 安装 AWS Copilot CLI。有关部署示例应用程序的信息，请参阅 使用 AWS Copilot CLI 部署示例 Amazon ECS 应用程序。AWS Copilot CLI 的其他文档可在 AWS Copilot 网站获得。

安装 AWS Copilot CLI

您可以使用 Homebrew 或通过以下步骤手动下载二进制文件来安装 AWS Copilot CLI。

使用 Homebrew

以下命令用于使用 Homebrew 在 macOS 或 Linux 系统上安装 AWS Copilot CLI。在安装之前，您应该安装 Homebrew 软件。有关更多信息，请参阅 Homebrew。

```
brew install aws/tap/copilot-cli
```

下载二进制文件

作为 Homebrew 的替代方案，您可以在 macOS、Windows 或 Linux 系统上手动安装 AWS Copilot CLI。在操作系统中使用以下命令下载二进制文件。macOS 和 Linux 示例还包括将执行权限应用于二进制文件的命令，并列出帮助菜单，以验证安装是否正常运行。

macOS

对于 macOS：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

对于 macOS ARM 系统：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Linux

对于 Linux x86（64 位）系统：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

对于 Linux ARM 系统：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Windows

使用 Powershell，运行以下命令：

```
New-Item -Path 'C:\copilot' -ItemType directory; `
  Invoke-WebRequest -OutFile 'C:\copilot\copilot.exe' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe
```

（可选）使用 PGP 签名验证手动安装的 AWS Copilot CLI

AWS Copilot CLI 可执行文件是使用 PGP 签名进行加密签名的。PGP 签名可用于验证 AWS Copilot CLI 可执行文件。通过以下步骤使用 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。

   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
```

Linux

使用您的 Linux 风格的程序包管理器安装 gpg。

Windows

从 GnuPG 网站下载 Windows 简单安装程序，并以管理员身份安装。安装 GnuPG 后，关闭并重新打开管理员 PowerShell。

有关更多信息，请参阅 GnuPG 下载。

2. 验证 GnuPG 路径是否已添加到您的环境路径中。

macOS

```
echo $PATH
```

如果在输出中没有看到 GnuPG 路径，请运行以下命令将其添加到路径中。

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

如果在输出中未看到 GnuPG 路径，则请运行以下命令，将其添加到路径中。

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

如果在输出中没有看到 GnuPG 路径，请运行以下命令将其添加到路径中。

```
$Env:PATH += ";<path to GnuPG executable files>"
```

3. 创建本地纯文本文件。

   macOS

   在终端上，输入：

   ```
   touch <public_key_filename.txt>
   ```

   使用 TextEdit 器打开文件。

   Linux

   在文本编辑器（如 gedit）中创建文本文件。另存为 `public_key_filename.txt`

   Windows

   在文本编辑器（如 Notepad）中创建文本文件。另存为 `public_key_filename.txt`

4. 添加 Amazon ECS PGP 公有密码的以下内容，保存此文件。

```
-----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-----
```

便于参考的 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
```

5. 在终端中使用以下命令导入带有 Amazon ECS PGP 公有密钥的文件。

```
gpg --import <public_key_filename.txt>
```

6. 下载 AWS Copilot 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`。

   macOS

   对于 macOS 系统，请使用以下命令。

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
   download/copilot-darwin.asc
   ```

Linux

对于 Linux x86（64 位）系统，请运行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux.asc
```

对于 Linux ARM 系统，请运行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-linux-arm64.asc
```

Windows

使用 Powershell，运行以下命令：

```
Invoke-WebRequest -OutFile 'C:\copilot\copilot.asc' https://github.com/aws/copilot-cli/releases/latest/download/copilot-windows.exe.asc
```

7. 运行以下命令验证签名。

- 对于 macOS 和 Linux 系统：

```
gpg --verify copilot.asc /usr/local/bin/copilot
```

- 对于 Windows 系统：

```
gpg --verify 'C:\copilot\copilot.asc' 'C:\copilot\copilot.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。

8. 对于 Windows 安装，请在 Powershell 上运行以下命令将 AWS Copilot 目录添加到路径中。

```
$Env:PATH += ";<path to Copilot executable files>"
```

使用 AWS Copilot CLI 部署示例 Amazon ECS 应用程序

安装 AWS Copilot CLI 后，您可以按照以下步骤部署示例应用程序、验证部署并清理资源。

先决条件

在开始之前，确保满足以下要求：

- 安装和配置 AWS CLI。有关更多信息，请参阅 AWS 命令行界面。
- 运行 `aws configure` 设置默认配置文件，该配置文件中的 AWS Copilot CLI 将用于托管您的应用程序和服务。
- 安装并运行 Docker 有关更多信息，请参阅 Docker 入门。

请使用单个命令部署示例 Amazon ECS 应用程序

1. 请使用以下命令部署从 GitHub 存储库克隆的示例 Web 应用程序。有关 AWS Copilot `init` 及其标记的更多信息，请参阅 AWS Copilot 文档。

```
git clone https://github.com/aws-samples/aws-copilot-sample-service.git demo-app &&
 \
cd demo-app &&                                \
copilot init --app demo                       \
  --name api                                  \
  --type 'Load Balanced Web Service'          \
  --dockerfile './Dockerfile'                 \
  --port 80                                   \
  --deploy
```

2. 部署完成后，AWS Copilot CLI 将返回一个可用于验证部署的 URL。您还可以使用以下命令验证应用的状态。

   - 列出您的所有 AWS Copilot 应用。

     ```
     copilot app ls
     ```

   - 显示有关应用程序中的环境和服务的信息。

     ```
     copilot app show
     ```

   - 显示有关您的环境的信息。

     ```
     copilot env ls
     ```

   - 显示有关服务的信息，包括终端、容量和相关资源。

     ```
     copilot svc show
     ```

   - 应用程序中所有服务的列表。

     ```
     copilot svc ls
     ```

   - 显示已部署服务的日志。

     ```
     copilot svc logs
     ```

   - 显示服务状态。

     ```
     copilot svc status
     ```

3. 完成本演示后，请运行以下命令清除相关资源，并避免因未使用的资源产生费用。

   ```
   copilot app delete
   ```

使用 AWS CDK 创建 Amazon ECS 资源

AWS Cloud Development Kit (AWS CDK) 是一个基础设施即代码 (IAC) 框架，可以让您使用所选编程语言来定义 AWS 云基础设施。要定义您自己的云基础设施，请首先编写一个包含一个或多个堆栈的应用程序（使用 CDK 支持的一种语言）。然后，将其合成为 AWS CloudFormation 模板并将您的资源

部署到 AWS 账户。按照本主题中的步骤，使用 Amazon Elastic Container Service (Amazon ECS) 和 Fargate 上的 AWS CDK 部署容器化 Web 服务器。

CDK 中包含的 AWS 构造库提供可用于对 AWS 服务 提供的资源进行建模的模块。对于常用的服务，该库提供具有智能默认值和最佳实践的精选构造。其中一个模块（特别是 `aws-ecs-patterns`）提供高级抽象，让您只需几行代码即可定义容器化服务和所有必要的支持资源。

本主题使用 `ApplicationLoadBalancedFargateService` 构造。此构造会在应用程序负载均衡器后面的 Fargate 上部署 Amazon ECS 服务。`aws-ecs-patterns` 模块还包括使用网络负载平衡器并在 Amazon EC2 上运行的构造。

在开始此任务之前，请设置您的 AWS CDK 开发环境，并通过运行以下命令安装 AWS CDK。有关如何设置 AWS CDK 开发环境的说明，请参阅 AWS CDK 入门 - 先决条件。

```
npm install -g aws-cdk
```

> **Note**
>
> 这些说明假设您在使用 AWS CDK v2。

主题

- 步骤 1：设置您的 AWS CDK 项目
- 步骤 2：使用 AWS CDK 在 Fargate 上定义容器化 Web 服务器
- 步骤 3：测试 Web 服务器
- 步骤 4：清除
- 后续步骤

步骤 1：设置您的 AWS CDK 项目

为您的新 AWS CDK 应用程序创建目录并初始化项目。

TypeScript

```
mkdir hello-ecs
cd hello-ecs
```

```
cdk init --language typescript
```

JavaScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language javascript
```

Python

```
mkdir hello-ecs
cd hello-ecs
cdk init --language python
```

项目启动后，请激活项目的虚拟环境并安装 AWS CDK 的基线依赖关系。

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir hello-ecs
cd hello-ecs
cdk init --language java
```

将此 Maven 项目导入到 Java IDE 中。例如，在 Eclipse 中，使用 File（文件）> Import（导入）> Maven > Existing Maven Projects（现有 Maven 项目）。

C#

```
mkdir hello-ecs
cd hello-ecs
cdk init --language csharp
```

Go

```
mkdir hello-ecs
cd hello-ecs
cdk init --language go
```

> **Note**
>
> AWS CDK 应用程序模板使用项目目录的名称来生成源文件和类的名称。在此示例中，该目录名为 `hello-ecs`。如果您使用其他项目目录名称，则您的应用将与这些说明不匹配。

AWS CDK v2 在名为 `aws-cdk-lib` 的单个程序包中包含适用于所有 AWS 服务 的稳定构造。当您初始化该项目时，此程序包作为依赖项进行安装。使用某些编程语言时，会在您首次构建项目时安装该程序包。本主题介绍如何使用 Amazon ECS 模式构造，它为使用 Amazon ECS 提供高级抽象。此模块依赖 Amazon ECS 构造和其他构造来预置您的 Amazon ECS 应用程序所需的资源。

将这些库导入 CDK 应用程序时使用的名称可能略有不同，具体取决于您使用的编程语言。下面提供了每种受支持的 CDK 编程语言中使用的名称，以方便参考。

TypeScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

JavaScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

Python

```
aws_cdk.aws_ecs
aws_cdk.aws_ecs_patterns
```

Java

```
software.amazon.awscdk.services.ecs
software.amazon.awscdk.services.ecs.patterns
```

C#

```
Amazon.CDK.AWS.ECS
Amazon.CDK.AWS.ECS.Patterns
```

Go

```
github.com/aws/aws-cdk-go/awscdk/v2/awsecs
github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns
```

步骤 2：使用 AWS CDK 在 Fargate 上定义容器化 Web 服务器

使用 DockerHub 中的容器映像 amazon-ecs-sample。此映像包含在 Amazon Linux 2 上运行的 PHP Web 应用。

在您创建的 AWS CDK 项目中，编辑包含堆栈定义的文件，使其类似于以下示例之一。

> **Note**
>
> 堆栈是一个部署单元。所有资源都必须位于一个堆栈中，并且一个堆栈中的所有资源都是同时部署的。如果某资源无法部署，则会回滚已部署的任何其他资源。一个 AWS CDK 应用可以包含多个堆栈，一个堆栈中的资源可以引用另一个堆栈中的资源。

TypeScript

更新 `lib/hello-ecs-stack.ts`，使其类似于以下内容。

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsp from 'aws-cdk-lib/aws-ecs-patterns';

export class HelloEcsStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}
```

JavaScript

更新 lib/hello-ecs-stack.js，使其类似于以下内容。

```
const cdk = require('aws-cdk-lib');
const { Construct } = require('constructs');
const ecs = require('aws-cdk-lib/aws-ecs');
const ecsp = require('aws-cdk-lib/aws-ecs-patterns');

class HelloEcsStack extends cdk.Stack {
  constructor(scope = Construct, id = string, props = cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}

module.exports = { HelloEcsStack }
```

Python

更新 hello-ecs/hello_ecs_stack.py，使其类似于以下内容。

```
import aws_cdk as cdk
from constructs import Construct

import aws_cdk.aws_ecs as ecs
import aws_cdk.aws_ecs_patterns as ecsp

class HelloEcsStack(cdk.Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ecsp.ApplicationLoadBalancedFargateService(self, "MyWebServer",
            task_image_options=ecsp.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry("amazon/amazon-ecs-sample")),
            public_load_balancer=True
```

```
        )
```

Java

更新 src/main/java/com.myorg/HelloEcsStack.java，使其类似于以下内容。

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;

import software.amazon.awscdk.services.ecs.ContainerImage;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedFargateService;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedTaskImageOptions;

public class HelloEcsStack extends Stack {
    public HelloEcsStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public HelloEcsStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        ApplicationLoadBalancedFargateService.Builder.create(this, "MyWebServer")
         .taskImageOptions(ApplicationLoadBalancedTaskImageOptions.builder()
           .image(ContainerImage.fromRegistry("amazon/amazon-ecs-sample"))
           .build())
         .publicLoadBalancer(true)
         .build();
    }
}
```

C#

更新 src/HelloEcs/HelloEcsStack.cs，使其类似于以下内容。

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.ECS;
```

```
using Amazon.CDK.AWS.ECS.Patterns;
namespace HelloEcs
{
    public class HelloEcsStack : Stack
    {
        internal HelloEcsStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            new ApplicationLoadBalancedFargateService(this, "MyWebServer",
                new ApplicationLoadBalancedFargateServiceProps
                {
                    TaskImageOptions = new ApplicationLoadBalancedTaskImageOptions
                    {
                        Image = ContainerImage.FromRegistry("amazon/amazon-ecs-
sample")
                    },
                    PublicLoadBalancer = true
                });
        }
    }
}
```

Go

更新 hello-ecs.go，使其类似于以下内容。

```
package main

import (
 "github.com/aws/aws-cdk-go/awscdk/v2"
 // "github.com/aws/aws-cdk-go/awscdk/v2/awssqs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns"
 "github.com/aws/constructs-go/constructs/v10"
 "github.com/aws/jsii-runtime-go"
)

type HelloEcsStackProps struct {
 awscdk.StackProps
}

func NewHelloEcsStack(scope constructs.Construct, id string, props
 *HelloEcsStackProps) awscdk.Stack {
 var sprops awscdk.StackProps
```

```
 if props != nil {
  sprops = props.StackProps
 }
 stack := awscdk.NewStack(scope, &id, &sprops)

 // The code that defines your stack goes here

 // example resource
 // queue := awssqs.NewQueue(stack, jsii.String("HelloEcsQueue"),
 &awssqs.QueueProps{
 //  VisibilityTimeout: awscdk.Duration_Seconds(jsii.Number(300)),
 // })
 res := awsecspatterns.NewApplicationLoadBalancedFargateService(stack,
 jsii.String("MyWebServer"),
  &awsecspatterns.ApplicationLoadBalancedFargateServiceProps{
   TaskImageOptions: &awsecspatterns.ApplicationLoadBalancedTaskImageOptions{
    Image: awsecs.ContainerImage_FromRegistry(jsii.String("amazon/amazon-ecs-
sample"), &awsecs.RepositoryImageProps{}),
   },
  },
 )
 awscdk.NewCfnOutput(stack, jsii.String("LoadBalancerDNS"),
 &awscdk.CfnOutputProps{Value: res.LoadBalancer().LoadBalancerDnsName()})

 return stack
}

func main() {
 defer jsii.Close()

 app := awscdk.NewApp(nil)

 NewHelloEcsStack(app, "HelloEcsStack", &HelloEcsStackProps{
  awscdk.StackProps{
   Env: env(),
  },
 })

 app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
```

```
func env() *awscdk.Environment {
 // If unspecified, this stack will be "environment-agnostic".
 // Account/Region-dependent features and context lookups will not work, but a
 // single synthesized template can be deployed anywhere.
 //---------------------------------------------------------------------------
 return nil

 // Uncomment if you know exactly what account and region you want to deploy
 // the stack to. This is the recommendation for production stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String("123456789012"),
 //  Region:  jsii.String("us-east-1"),
 // }

 // Uncomment to specialize this stack for the AWS Account and Region that are
 // implied by the current CLI configuration. This is recommended for dev
 // stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
 //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
 // }
}
```

前面的简短片段包括以下内容：

- 服务的逻辑名称：`MyWebServer`。
- 从 DockerHub 获取的容器映像：`amazon/amazon-ecs-sample`。
- 其他相关信息，例如负载均衡器有公有地址并且可以从互联网访问这一事实。

AWS CDK 将创建部署 Web 服务器所需的所有资源，包括以下资源。此示例中省略了这些资源。

- Amazon ECS 集群
- Amazon VPC 和 Amazon EC2 实例
- 自动扩缩组
- 应用程序负载均衡器
- IAM 角色和策略

一些自动预置的资源由堆栈中定义的所有 Amazon ECS 服务共享。

保存源文件，然后在应用程序的主目录中运行 `cdk synth` 命令。AWS CDK 运行应用程序并从中合成一个 AWS CloudFormation 模板，然后显示该模板。该模板是一个约 600 行的 YAML 文件。此处显示了文件的开头。您的模板可能与此示例有所不同。

```
Resources:
  MyWebServerLB3B5FD3AB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      LoadBalancerAttributes:
        - Key: deletion_protection.enabled
          Value: "false"
      Scheme: internet-facing
      SecurityGroups:
        - Fn::GetAtt:
            - MyWebServerLBSecurityGroup01B285AA
            - GroupId
      Subnets:
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1Subnet3C273B99
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2Subnet95FF715A
      Type: application
    DependsOn:
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1DefaultRouteFF4E2178
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2DefaultRouteB1375520
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/Resource
  MyWebServerLBSecurityGroup01B285AA:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Automatically created Security Group for ELB
 HelloEcsStackMyWebServerLB06757F57
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          Description: Allow from anyone on port 80
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
      VpcId:
        Ref: EcsDefaultClusterMnL3mNNYNVpc7788A521
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/SecurityGroup/Resource
```

```
# and so on for another few hundred lines
```

要在您的 AWS 账户 中部署服务，请在应用程序的主目录中运行 `cdk deploy` 命令。将要求您批准 AWS CDK 生成的 IAM policy。

部署需要几分钟，在此期间，AWS CDK 将创建多个资源。部署输出的最后几行包括负载均衡器的公共主机名和新 Web 服务器的 URL。它们如下所示。

```
Outputs:
HelloEcsStack.MyWebServerLoadBalancerDNSXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
```

步骤 3：测试 Web 服务器

从部署输出复制 URL 并将其粘贴到 Web 浏览器。将显示来自 Web 服务器的以下欢迎消息。

步骤 4：清除

完成 Web 服务器的使用后，通过在应用程序的主目录中运行 `cdk destroy` 命令，使用 CDK 结束服务。这样做可以防止您在未来产生任何意外费用。

后续步骤

要了解有关如何使用 AWS CDK 开发 AWS 基础设施的更多信息，请参阅《AWS CDK 开发人员指南》。

有关使用所选语言编写 AWS CDK 应用程序的信息，请参阅以下内容：

TypeScript

在 TypeScript 中使用 AWS CDK

JavaScript

在 JavaScript 中使用 AWS CDK

Python

在 Python 中使用 AWS CDK

Java

在 Java 中使用 AWS CDK

C#

在 C# 中使用 AWS CDK

Go

在 Go 中使用 AWS CDK

有关本主题中使用的 AWS 构造库模块的更多信息，请参阅以下 AWS CDK API 参考概述。

- aws-ecs
- aws-ecs-patterns

使用 AWS CloudFormation 创建 Amazon ECS 资源

Amazon ECS 与 AWS CloudFormation 集成，该服务可用于使用您定义的模板对 AWS 资源进行建模和设置。这样，您可以花费更少的时间来创建和管理您的资源和基础设施。使用 AWS CloudFormation，您可以创建一个模板来描述您想要的所有 AWS 资源，例如特定的 Amazon ECS 集群。然后，AWS CloudFormation 会负责为您预置和配置这些资源。

当您使用 AWS CloudFormation 时，您可以重复使用您的模板以一致且可重复的方式设置您的 Amazon ECS 资源。您仅描述您的资源一次，然后跨多个 AWS 账户 和 AWS 区域 再次预置相同的资源。

AWS CloudFormation 模板

要为 Amazon ECS 和相关服务预置和配置资源，请确保您熟悉 AWS CloudFormation 模板。AWS CloudFormation 模板是 JSON 或者 YAML 格式的文本文件，描述您要在 AWS CloudFormation

堆栈中预置的资源。如果您不熟悉 JSON 或 YAML 格式，或两者都不熟悉，则可以使用 AWS CloudFormation Designer 开始使用 AWS CloudFormation 模板。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的什么是 AWS CloudFormation Designer？。

Amazon ECS支持在 AWS CloudFormation 中创建集群、任务定义、服务和任务集。以下示例演示如何使用 AWS CLI 通过这些模板创建资源。您也可以使用 AWS CloudFormation 控制台创建这些资源。有关如何使用 AWS CloudFormation 控制台创建资源的更多信息，请参阅《AWS CloudFormation 用户指南》。

示例模板

使用单独的堆栈创建 Amazon ECS 资源

以下示例介绍如何通过对每个资源使用单独的堆栈来创建 Amazon ECS 资源。

任务定义

您可以使用以下模板来创建 Fargate Linux 任务。

JSON

```
 {
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
        "ECSTaskDefinition": {
           "Type": "AWS::ECS::TaskDefinition",
           "Properties": {
               "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::aws_account_id:role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }

        }
        }
    }
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
```

```
  Properties:
    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::aws_account_id:role/ecsTaskExecutionRole'
    Family: task-definition-cfn
    Memory: 512
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX
```

集群

您可以使用以下模板来创建空集群。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "MyEmptyCluster"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: MyEmptyCluster
```

在一个堆栈中创建多个 Amazon ECS 资源

您可以使用以下示例模板来在一个堆栈中创建多个 Amazon ECS 资源。该模板将创建名为 `CFNCluster` 的 Amazon ECS 集群。该集群包含设置 Web 服务器的 Linux Fargate 任务定义。该模板还会创建一个名为 `cfn-service` 的服务，其将启动并维护任务定义所定义的任务。在使用此模板之前，请确保服务的 `NetworkConfiguration` 中的子网和安全组 ID 全部都属于同一 VPC，且安全组具有必要的规则。有关安全组规则的更多信息，请参阅《Amazon VPC 用户指南》中的安全组规则。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "CFNCluster"
```

```
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "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::aws_account_id::role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
```

```
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }
        },
        "ECSService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "ServiceName": "cfn-service",
                "Cluster": {
                    "Ref": "ECSCluster"
                },
                "DesiredCount": 1,
                "LaunchType": "FARGATE",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "AssignPublicIp": "ENABLED",
                        "SecurityGroups": [
                            "sg-abcdef01234567890"
                        ],
                        "Subnets": [
                            "subnet-abcdef01234567890"
                        ]
                    }
                },
                "TaskDefinition": {
                    "Ref": "ECSTaskDefinition"
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
```

```
  Properties:
    ClusterName: CFNCluster
ECSTaskDefinition:
  Type: 'AWS::ECS::TaskDefinition'
  Properties:
    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::aws_account_id:role/ecsTaskExecutionRole'
    Family: task-definition-cfn
    Memory: 512
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX
ECSService:
  Type: 'AWS::ECS::Service'
  Properties:
    ServiceName: cfn-service
```

```
      Cluster: !Ref ECSCluster
      DesiredCount: 1
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - sg-abcdef01234567890
          Subnets:
            - subnet-abcdef01234567890
      TaskDefinition: !Ref ECSTaskDefinition
```

使用 AWS CLI 从模板创建资源

以下命令将使用名为 ecs-template-body.json 的模板正文文件创建名为 ecs-stack 的堆栈。确保模板正文文件为 JSON 或 YAML 格式。文件的位置在 --template-body 参数中指定。在这种情况下，模板正文文件位于当前目录中。

```
aws cloudformation create-stack \
      --stack-name ecs-stack \
      --template-body file://ecs-template-body.json
```

要确保正确创建资源，请检查 Amazon ECS 控制台或者使用以下命令：

- 以下命令将列出所有任务定义。

  ```
  aws ecs list-task-definitions
  ```

- 以下命令将列出所有集群。

  ```
  aws ecs list-clusters
  ```

- 以下命令将列出集群 *CFNCluster* 中定义的所有服务。将 *CFNCluster* 替换为您要在其中创建服务的集群的名称。

  ```
  aws ecs list-services \
        --cluster CFNCluster
  ```

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation 命令行界面用户指南

开始使用 Amazon ECS 命令行界面

Amazon ECS 发布了 AWS Copilot，这是一种命令行界面（CLI）工具，可简化在本地开发环境中在Amazon ECS上构建、发布和操作生产就绪的容器化应用程序。有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。

Amazon Elastic Container Service（Amazon ECS）命令行界面（CLI）提供高级命令，以简化本地开发环境中集群和任务的创建、更新和监控。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 Management Console。

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。

GitHub 上提供了 Amazon ECS CLI 的源代码。该工具不再主动开发。

安装 Amazon ECS CLI

Amazon ECS 发布了 AWS Copilot，这是一种命令行界面（CLI）工具，可简化在本地开发环境中在Amazon ECS上构建、发布和操作生产就绪的容器化应用程序。有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。

以下步骤说明如何在 macOS、Linux 或 Windows 系统上安装 Amazon ECS CLI。

安装 Amazon ECS CLI

1. 下载 Amazon ECS CLI 二进制文件。

macOS

```
sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/
ecs-cli-darwin-amd64-latest
```

Linux

```
sudo curl -Lo /usr/local/bin/ecs-cli https://amazon-ecs-cli.s3.amazonaws.com/
ecs-cli-linux-amd64-latest
```

Windows

打开 Windows PowerShell 并输入以下命令。

> **Note**
>
> 如果您遇到权限问题，请确保您在 Windows 上具有管理员访问权限，并且以管理员身份运行 PowerShell。

```
New-Item -Path 'C:\Program Files\Amazon\ECSCLI' -ItemType Directory
Invoke-WebRequest -OutFile 'C:\Program Files\Amazon\ECSCLI\ecs-cli.exe' https://
amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe
```

2. 使用 PGP 签名验证 Amazon ECS CLI。Amazon ECS CLI 可执行文件是使用 PGP 签名进行加密签名的。PGP 签名可用于验证 Amazon ECS CLI 可执行文件的有效性。通过以下步骤使用 GnuPG 工具验证签名。

   a. 下载并安装 GnuPG。有关更多信息，请参阅 GnuPG 网站。

      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
      ```

Linux

使用您的 Linux 风格的程序包管理器安装 gpg。

Windows

从 GnuPG 网站下载 Windows 简单安装程序，并以管理员身份安装。安装 GnuPG 后，关闭并重新打开管理员 PowerShell。

有关更多信息，请参阅 GnuPG 下载。

b. 验证 GnuPG 路径是否已添加到您的环境路径中。

macOS

```
echo $PATH
```

如果在输出中没有看到 GnuPG 路径，请运行以下命令将其添加到路径中。

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

如果在输出中未看到 GnuPG 路径，则请运行以下命令，将其添加到路径中。

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

如果在输出中没有看到 GnuPG 路径，请运行以下命令将其添加到路径中。

```
$Env:PATH += ";<path to GnuPG executable files>"
```

c. 创建本地纯文本文件。

macOS

在终端上，输入：

```
touch <public_key_filename.txt>
```

使用 TextEdit 器打开文件。

Linux

在文本编辑器（如 gedit）中创建文本文件。另存为 `public_key_filename.txt`

Windows

在文本编辑器（如 Notepad）中创建文本文件。另存为 `public_key_filename.txt`

d. 添加 Amazon ECS PGP 公有密码的以下内容，保存此文件。

```
-----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-----
```

便于参考的 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
```

e. 在终端中使用以下命令导入带有 Amazon ECS PGP 公有密钥的文件。

```
gpg --import <public_key_filename.txt>
```

f. 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`。

macOS

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-darwin-amd64-latest.asc
```

Linux

```
curl -Lo ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-linux-amd64-latest.asc
```

Windows

```
Invoke-WebRequest -OutFile ecs-cli.asc https://amazon-ecs-cli.s3.amazonaws.com/ecs-cli-windows-amd64-latest.exe.asc
```

g. 验证签名。

macOS and Linux

```
gpg --verify ecs-cli.asc /usr/local/bin/ecs-cli
```

Windows

```
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。

3. 将执行权限应用于二进制文件。

macOS and Linux

```
sudo chmod +x /usr/local/bin/ecs-cli
```

Windows

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：

```
setx path "%path%;C:\Program Files\Amazon\ECSCLI"
```

重新启动 PowerShell，以便更改生效。

> **Note**
>
> 设置 PATH 变量后，可以从 Windows PowerShell 或命令提示符使用 Amazon ECS CLI。

4. 验证 CLI 是否正常运行。

```
ecs-cli --version
```

继续执行配置 Amazon ECS CLI。

> **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。有关更多信息，请参阅 配置 Amazon ECS CLI。

配置 Amazon ECS CLI

Amazon ECS 发布了 AWS Copilot，这是一种命令行界面（CLI）工具，可简化在本地开发环境中在Amazon ECS上构建、发布和操作生产就绪的容器化应用程序。有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。

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* 替换为预设情况下要使用的任务启动类型、*region_name* 替换为所需的 AWS 区域、*cluster_name* 替换为要使用的现有 Amazon ECS 集群或新集群的名称，并将 *configuration_name* 替换为您希望提供此配置的名称。

   ```
   ecs-cli configure --cluster cluster_name --default-launch-type launch_type --
   region region_name --config-name configuration_name
   ```

使用配置文件

Amazon ECS CLI 支持使用 ecs-cli configure profile 命令将多组 AWS 凭据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。

使用集群配置

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` 选项配置多个命名群集配置。

了解优先顺序

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. Amazon 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 配置文件中获取凭据。
4. 原定设置 AWS 配置文件-尝试使用 AWS 配置文件名称中的凭据 (`aws_access_key_id`, `aws_secret_access_key`) 或 `assume_role` (`role_arn`, `source_profile`)。
   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. Environment variables - 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`)

适用于 Amazon ECS 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，您不必再预配置、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，您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 policy 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。通过将 `requiresCompatibilities` 任务定义参数设置为 `FARGATE`，您可以为 Fargate 配置任务定义。有关更多信息，请参阅 启动类型。

Fargate 为 Amazon Linux 2 以及 Microsoft Windows 2019 Server Full 和 Core 版本提供平台版本。除非另有说明，否则本页面上的信息适用于所有 Fargate 平台。

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。

有关在 Fargate 上支持 Linux 容器的区域的信息，请参阅 the section called “AWS Fargate 上的 Linux 容器”。

有关在 Fargate 上支持 Windows 容器的区域的信息，请参阅 the section called “AWS Fargate 上的 Windows 容器”。

演练

有关如何开始使用控制台的信息，请参阅：

- 了解如何创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 了解如何创建 Fargate 启动类型的 Amazon ECS Windows 任务

有关如何开始使用 AWS CLI 的信息，请参阅：

- 使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Windows 任务

容量提供程序

提供以下容量提供程序：

- Fargate
- Fargate Spot – 按照与 AWS Fargate 价格相比的折扣价格运行能够容忍中断的 Amazon ECS 任务。Fargate Spot 在备用计算容量上运行任务。当 AWS 需要恢复容量时，您的任务将被中断，并发出两分钟的警告。有关更多信息，请参阅 Fargate 启动类型的 Amazon ECS 集群 。

  您只能将 Fargate Spot 用于使用 X86 架构的 Linux 任务。

任务定义

使用 Fargate 启动类型的任务并非支持所有可用的 Amazon ECS 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。有关更多信息，请参阅 任务 CPU 和内存。

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。在运行任务或创建维护多个相同任务的服务时，您可以选择平台版本。

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，将会发布新的平台版本修订版。Fargate 平台版本通过进行新的平台版本修订来更新。每个任务在其生命周期内都在一个平台版本修订版上运行。如果您想使用最新的平台版本修订版，则必须启动新任务。在 Fargate 上运行的新任务始终在平台版本的最新修订版上运行，从而确保了任务始终在安全的、经过修补的基础设施上启动。

如果发现影响现有平台版本的安全问题，AWS 将为该平台版本创建新的补丁修订版，并停止在有漏洞的修订版上运行的任务。在某些情况下，您可能会收到计划停止您在 Fargate 上的任务的通知。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate 任务维护常见问题。

有关更多信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本和适用于 Amazon ECS 的 Fargate Windows 平台版本。

服务负载均衡

您可以选择将 AWS Fargate 上的 Amazon ECS 服务配置为使用 Elastic Load Balancing 平均分配服务中的任务流量。

AWS Fargate 上的 Amazon ECS 服务支持 Application Load Balancer 和 Network Load Balancer 负载均衡器类型。Application Load Balancers 用于路由 HTTP/HTTPS (或第 7 层) 流量。Network

Load Balancers 用于路由 TCP 或 UDP（或第 4 层）流量。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。

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 使用负载均衡分配 Amazon ECS 服务流量。

只有在使用平台版本 1.4 或更高版本时，才支持使用 Network Load Balancer 将 UDP 流量路由到 AWS Fargate 任务上的 Amazon ECS。

使用情况指标

您可以使用 CloudWatch 用量指标来提供账户资源使用情况的可见性。这些指标可在 CloudWatch 图表和控制面板上直观呈现当前的服务使用情况。

AWS Fargate 用量指标与 AWS 服务配额对应。您可以配置警报，以在用量接近服务限额时向您发出警报。有关 AWS Fargate 服务配额的更多信息，请参阅 AWS Fargate 服务限额。

有关 AWS Fargate 使用情况指标的更多信息，请参阅 AWS Fargate 的 Amazon Elastic Container Service 用户指南中的 AWS Fargate 使用情况指标。

了解何时使用 Fargate 启动类型的 Amazon ECS 安全注意事项

建议为其任务寻求强隔离的客户使用 Fargate。Fargate 在硬件虚拟化环境中运行每个任务。这将确保这些容器化工作负载不会与其他任务共享网络接口、Fargate 临时存储、CPU 或内存。有关更多信息，请参阅 AWS Fargate 的安全概述。

Amazon ECS 中的 Fargate 安全最佳实践

在使用 AWS Fargate 时，建议您考虑以下最佳实践。有关其他指南，请参阅 AWS Fargate 的安全性概述。

使用 AWS KMS 加密 Fargate 的临时存储

您应该具有 AWS KMS 加密的临时存储。对于使用平台版本 `1.4.0` 或更高版本在 Fargate 上托管的任务，每个任务会获得 20 GiB 的临时存储。您可以增加临时存储总量，最多可达 200GiB，方法是在您的任务定义中指定 `ephemeralStorage` 参数。对于在 2020 年 5 月 28 日或之后启动的此类任务，将使用 AES-256 加密算法以及由 Fargate 托管的加密密钥对临时存储进行加密。

有关更多信息，请参阅在任务中使用数据卷。

示例：使用临时存储加密在 Fargate 平台版本 1.4.0 上启动任务

以下命令将在 Fargate 平台版本 1.4 上启动任务。由于此任务是作为集群的一部分启动的，因此会使用自动加密的 20 GiB 临时存储。

```
aws ecs run-task --cluster clustername \
  --task-definition taskdefinition:version \
  --count 1
  --launch-type "FARGATE" \
  --platform-version 1.4.0 \
  --network-configuration
 "awsvpcConfiguration={subnets=[subnetid],securityGroups=[securitygroupid]}" \
  --region region
```

使用 Fargate 进行内核系统调用跟踪的 SYS_PTRACE 功能

添加或从容器中移除的 Linux 功能的默认配置由 Docker 提供。有关可用功能的更多信息，请参阅 Docker 运行文档中的运行时系统权限和 Linux 功能。

在 Fargate 上启动的任务仅支持添加 SYS_PTRACE 内核功能。

下面的教程视频展示了如何通过 Sysdig Falco 项目使用此功能。

#ContainersFromTheCouch – 使用 SYS_PTRACE 功能排查 Fargate 任务问题

上一个视频中讨论的代码可以在 GitHub 上在此处找到。

将 Amazon GuardDuty 与 Fargate 运行时监控结合使用

Amazon GuardDuty 是一项威胁检测服务，可帮助保护您的账户、容器、工作负载和 AWS 环境中的数据。GuardDuty 使用机器学习（ML）模型以及异常和威胁检测功能，持续监控不同的日志源和运行时活动，以识别环境中的潜在安全风险和恶意活动并确定其优先级。

GuardDuty 中的运行时监控通过持续监控 AWS 日志和联网活动来识别恶意或未经授权的行为，从而保护在 Fargate 上运行的工作负载。运行时监控使用轻量级、完全托管的 GuardDuty 安全代理分析主机中的行为，例如文件访问、进程执行和网络连接。它涉及的问题包括权限升级、使用公开的凭证，或与恶意 IP 地址、域通信，以及您的 Amazon EC2 实例和容器工作负载上存在恶意软件。有关更多信息，请参阅《GuardDuty User Guide》中的 GuardDuty Runtime Monitoring。

Amazon ECS 的 Fargate 安全注意事项

每项任务都有专用的基础设施容量，因为 Fargate 在隔离的虚拟环境中运行每个工作负载。在 Fargate 上运行的工作负载不与其他任务共享网络接口、临时存储、CPU 或内存。您可以在一个任务中运行多个容器，包括应用程序容器和 sidecar 容器，或者仅仅是 sidecar。sidecar 是在 Amazon ECS 任务中与应用程序容器一起运行的容器。当应用程序容器运行核心应用程序代码时，在 sidecar 中运行的进程可以增强应用程序。Sidecar 可帮助您将应用程序功能隔离到专用容器中，从而更轻松地更新应用程序的各个部分。

属于同一任务的容器会共享 Fargate 启动类型的资源，因为这些容器将始终在同一台主机上运行，并共享计算资源。这些容器还共享 Fargate 提供的临时存储空间。任务中的 Linux 容器共享网络命名空间，包括 IP 地址和网络端口。在任务中，属于该任务的容器可以通过 localhost 进行相互通信。

Fargate 中的运行时系统环境会阻止您使用 EC2 实例支持的特定控制器功能。设计在 Fargate 上运行的工作负载时，请考虑以下事项：

- 没有特权容器或访问权限 - Fargate 目前不提供诸如特权容器或访问权限之类的功能。这将影响应用场景，例如在 Docker 中运行 Docker。
- 对 Linux 功能的访问受限 - 容器在 Fargate 上运行的环境已被锁定。其他 Linux 功能（例如 CAP_SYS_ADMIN 和 CAP_NET_ADMIN）受到限制，以防止权限升级。Fargate 支持向任务添加 CAP_SYS_PTRACE Linux 功能，以允许在任务中部署的可观测性和安全工具来监控容器化应用程序。
- 无法访问底层主机 - 客户和 AWS 操作人员都无法连接到运行客户工作负载的主机。您可以使用 ECS Exec 在 Fargate 上运行的容器中运行命令或获取 shell。您可以使用 ECS exec 来帮助收集用于调试的诊断信息。Fargate 还阻止容器访问底层主机的资源，例如文件系统、设备、联网和容器运行时系统。
- 联网 - 您可以使用安全组和网络 ACL 来控制入站和出站流量。Fargate 任务会从您的 VPC 中配置的子网接收一个 IP 地址。

适用于 Amazon ECS 的 Fargate Linux 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。在运行任务或创建维护多个相同任务的服务时，您可以选择平台版本。

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，将会发布新的平台版本修订版。Fargate 平台版本通过进行新的平台版本修订来更新。每个任务在其生命周期内都在一个平台版本修订版上运行。如果您想使用最新的平台版本修订版，则必须启动新任务。在

Fargate 上运行的新任务始终在平台版本的最新修订版上运行，从而确保了任务始终在安全的、经过修补的基础设施上启动。

如果发现影响现有平台版本的安全问题，AWS 将为该平台版本创建新的补丁修订版，并停止在有漏洞的修订版上运行的任务。在某些情况下，您可能会收到计划停止您在 Fargate 上的任务的通知。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate 任务维护常见问题。

注意事项

指定平台版本时请考虑以下事项：

- 指定平台版本时，您可以使用特定版本号，例如 `1.4.0` 或 `LATEST`。

  选择最新平台版本时，将使用 `1.4.0` 平台版本。
- 如果要更新某个服务的平台版本，请创建部署。例如，假设您有一个在 Linux 平台版本 `1.3.0` 上运行任务的服务。要将该服务改为在 Linux 平台版本 `1.4.0` 上运行任务，您可以更新该服务并指定新的平台版本。您的任务将使用最新的平台版本和最新的平台版本修订版重新部署。有关部署的更多信息，请参阅 Amazon ECS 服务。
- 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。例如，假设您有一个在 Linux 平台版本 `1.3.0` 上运行任务的服务。如果增加该服务的预期任务数，则服务调度器将使用平台版本 `1.3.0` 的最新平台版本修订版启动新任务。
- 新任务始终在平台版本的最新修订版上运行，从而确保了任务始终在安全的、经过修补的基础设施上启动。
- Fargate 上用于 Linux 容器和 Windows 容器的平台版本号是独立的。例如，Fargate 上的 Windows 容器的平台版本 `1.0.0` 中使用的行为、功能和软件无法与 Fargate 上的 Linux 容器的平台版本 `1.0.0` 相比。

下面是可用的 Linux 平台版本。有关平台版本弃用的信息，请参阅 AWS Fargate Linux 平台版本弃用。

1.4.0

以下是平台版本的更新日志 `1.4.0`。

- 自 2020 年 11 月 5 日起，任何Fargate 上使用平台版本 `1.4.0` 启用的新的 Amazon ECS 任务都能够使用以下功能：
  - 当使用 Secrets Manager 存储敏感数据时，您可以将特定 JSON 键或特定版本的密钥注入为环境变量或注入到日志配置中。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。

  - 批量指定环境变量，使用 `environmentFiles` 容器定义参数。有关更多信息，请参阅 将单个环境变量传递给 Amazon ECS 容器。
  - 在 VPC 和为 IPv6 启用的子网中运行的任务将同时分配一个专用 IPv4 地址和 IPv6 地址。有关更多信息，请参阅 AWS Fargate Amazon Elastic Container Service 开发人员指南 中的 Fargate任务联网。
  - 任务元数据端点版本 4 提供了有关您的任务和容器的其他元数据，包括任务启动类型、容器的 Amazon Resource Name (ARN) 以及使用的日志驱动程序和日志驱动程序选项。当查询 `/stats` 端点时，您还可以接收容器的网络速率统计数据。有关更多信息，请参阅任务元数据端点版本 4。
- 自 2020 年 7 月 30 日起，任何 Fargate 上使用平台版本 `1.4.0` 启用的新的 Amazon ECS 任务能够在 Fargate 任务中使用 Network Load Balancer 将 UDP 流量路由到他们的 Amazon ECS。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。
- 从 2020 年 5 月 28 日开始，任何 Fargate 上使用平台版本 `1.4.0` 启用的新的 Amazon ECS 任务都将使用 AES-256 加密算法通过 AWS 托管加密密钥来加密其短暂存储空间。有关更多信息，请参阅适用于 Amazon ECS 的 Fargate 临时存储 和Amazon ECS 任务的存储选项。
- 增加了对将 Amazon EFS 文件系统卷用于持久性任务存储的支持。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。
- 每个任务的短暂任务存储空间已增加到至少 20 GB。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate 临时存储。
- 针对任务和来自任务的网络流量行为已更新。从平台版本 1.4.0 开始，所有 Fargate 任务都会接收单个弹性网络接口（称为任务 ENI），所有网络流量都将流经 VPC 内的这个 ENI，并将通过 VPC 流日志对您可见。有关 Amazon EC2 启动类型联网的更多信息，请参阅 Fargate 任务联网。有关 Fargate 启动类型联网的更多信息，请参阅Fargate 启动类型的 Amazon ECS 任务联网选项。
- 任务 ENI 增加对巨型帧的支持。网络接口配置了最大传输单元 (MTU)，这是单个帧内将放入的最大负载的大小。MTU 越大，单个帧内可以放入的应用程序负载就越多，这可以减少每帧开销并提高效率。当您的任务和目标之间的网络路径支持巨型帧时，支持巨型帧将减少开销，如保留在您的 VPC 中的所有流量。
- CloudWatch Container Insights 将包括 Fargate 任务的网络性能指标。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。
- 增加了对任务元数据端点版本 4 的支持，该端点为您的 Fargate 任务提供附加信息，包括任务的网络统计信息以及任务所运行的可用区。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4 和 Fargate 上任务的 Amazon ECS 任务元数据端点版本 4。
- 增加了对容器定义中的 `SYS_PTRACE` Linux 参数的支持。有关更多信息，请参阅 Linux 参数。
- Amazon ECS 容器代理替代了对所有 Fargate 任务使用 Amazon ECS 容器代理。通常，此更改不会影响您的任务运行方式。

- 容器运行时现在使用 Containerd 而不是 Docker。此更改很可能不会影响您的任务运行方式。您会注意到，源自容器运行时的一些错误消息将从提及 Docker 变为更一般的错误。有关更多信息，请参阅 Amazon Elastic Container Service 用户指南AWS Fargate中的已停止的任务错误代码。
- 基于 Amazon Linux 2

1.3.0

以下是平台版本的更新日志 `1.3.0`。

- 从 2019 年 9 月 30 日开始，所启动的任何新的 Fargate 任务均支持 `awsfirelens` 日志驱动程序。配置 FireLens for Amazon ECS，使用任务定义参数将日志路由到 AWS 服务或 AWS 合作伙伴网络（APN）目标，以进行日志存储和分析。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。
- 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 Amazon Elastic Container Service 开发人员指南AWS Fargate中的任务维护。
- 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、容器依赖项 和 容器超时。
- 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 将敏感数据传递给 Amazon ECS 容器。
- 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 将敏感数据传递给 Amazon ECS 容器。
- 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。
- 从 2019 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 使用 Container Insights 监控 Amazon ECS 容器。
- 从 2019 年 12 月 3 日开始，支持 Fargate Spot 容量提供程序。有关更多信息，请参阅 Fargate 启动类型的 Amazon ECS 集群 。
- 基于 Amazon Linux 2

迁移到 Linux 平台版本 1.4.0

将 Fargate 任务上的 Amazon ECS 从平台版本 `1.0.0`、`1.1.0`、`1.2.0` 或 `1.3.0` 迁移到平台版本 `1.4.0` 时，请考虑以下事项。在迁移任务前，最好先确认您的任务在平台版本 `1.4.0` 上可以正常运行。

- 针对任务和来自任务的网络流量行为已更新。从平台版本 1.4.0 开始，Fargate 任务上的所有 Amazon ECS 都会接收单个弹性网络接口（称为任务 ENI），所有网络流量都将流经 VPC 内的这个 ENI，并将通过 VPC 流日志对您可见。有关更多信息，请参阅Fargate 启动类型的 Amazon ECS 任务联网选项。
- 如果您使用的是接口 VPC 端点，请考虑以下事项。
  - 使用 Amazon ECR 托管的容器映像时，需要 com.amazonaws.region.ecr.dkr 和 com.amazonaws.region.ecr.api VPC 端点以及 Amazon S3 网关端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。
  - 当使用引用 Secrets Manager 密钥来检索容器的敏感数据的任务定义时，您必须为 Secrets Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的将 Secrets Manager 与 VPC 端点结合使用。
  - 当使用引用 Systems Manager Parameter Store 参数来检索容器的敏感数据的任务定义时，您必须为 Systems Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Systems Manager 用户指南中的将 Systems Manager 用于 VPC 端点。
  - 确保与您的任务关联的弹性网络接口 (ENI) 中的安全组已创建安全组规则，以允许任务与您正在使用的 VPC 终端节点之间进行流量传输。

AWS Fargate Linux 平台版本弃用

此页面列出了 AWS Fargate 已弃用或已计划弃用的 Linux 平台版本。在到达公布的弃用日期之前，这些平台版本将仍然可用。

为计划弃用的每个平台版本提供强制更新日期。在强制更新日期，任何使用 `LATEST` 平台版本且指向计划弃用的平台版本的服务都将使用强制新部署选项进行更新。使用强制新部署选项更新服务时，在计划弃用的平台版本上运行的所有任务都将停止，并且新任务将使用 `LATEST` 标记指向。具有显式平台版本集的独立任务或服务不受强制更新日期的影响。

我们建议更新您的服务独立任务，使用最新的平台版本。有关迁移到最新平台版本的详细信息，请参阅迁移到 Linux 平台版本 1.4.0。

一旦平台版本到达了弃用日期，平台版本将不再可用于新任务或服务。任何明确使用已弃用的平台版本的独立任务或服务将继续使用该平台版本，直到任务停止为止。弃用日期后，已弃用的平台版本将不再接收任何安全更新或错误修复。

平台版本	强制更新日期	弃用日期
1.0.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.1.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.2.0	2020 年 10 月 26 日	2020 年 12 月 14 日

有关最新平台版本的信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。

已弃用的 Fargate AWS Linux 版本的更改日志

1.2.0

以下是平台版本的更新日志 `1.2.0`。

> **Note**
>
> 平台版本 `1.2.0` 不再可用。有关平台版本弃用的信息，请参阅 AWS Fargate Linux 平台版本弃用。

- 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 在 Amazon ECS 中使用非 AWS 容器映像。

1.1.0

以下是平台版本的更新日志 `1.1.0`。

> **Note**
>
> 平台版本 `1.1.0` 不再可用。有关平台版本弃用的信息，请参阅 AWS Fargate Linux 平台版本弃用。

- 添加了对 Amazon ECS 任务元数据端点的支持。有关更多信息，请参阅 Amazon ECS 任务元数据可用于 Fargate 上的任务。
- 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。
- 添加了对 Amazon ECS 服务发现的支持。有关更多信息，请参阅 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。

1.0.0

以下是平台版本的更新日志 `1.0.0`。

> **Note**
>
> 平台版本 `1.0.0` 不再可用。有关平台版本弃用的信息，请参阅 AWS Fargate Linux 平台版本弃用。

- 基于 Amazon Linux 2017.09
- 首次发布。

Fargate 上的 Linux 容器的 Amazon ECS 容器映像拉取行为

每个 Fargate 任务都在自己的单用途、单租户实例上运行。当您在 Fargate 上运行 Linux 容器时，容器映像或容器映像层不会缓存在实例上。因此，对于任务中定义的每个容器映像，需要从每个 Fargate 任务的容器映像注册表中提取整个容器映像。提取映像所需的时间与启动 Fargate 任务所花费的时间直接相关。

要优化映像提取时间，请考虑以下因素。

容器映像近似

要缩短下载容器映像所需的时间，请将数据放置在尽可能靠近计算机的位置。通过互联网或跨 AWS 区域 提取容器映像可能会影响下载时间。建议您将容器映像存储在运行任务的同一区域。如果您将容器映像存储在 Amazon ECR 中，则请使用 VPC 接口端点来进一步缩短映像提取时间。有关更多信息，请参阅《Amazon ECR 用户指南》中的 Amazon ECR 接口 VPC 端点（AWS PrivateLink）。

缩小容器映像大小

容器映像的大小直接影响下载时间。减小容器映像的大小或容器映像层的数量可以缩短映像下载所需的时间。轻量级基础映像（例如最小的 Amazon Linux 2023 容器映像）可能比基于传统操作系统基础映像的映像小得多。有关最小映像的更多信息，请参阅《Amazon Linux 2023 用户指南》中的 AL2023 最小容器映像。

替代压缩算法

推送到容器映像注册表时，容器映像层通常会被压缩。压缩容器映像层可以减少必须通过网络传输并存储在容器映像注册表中的数据量。在容器运行时将容器映像层下载到实例后，将对该层进行解压缩。使用的压缩算法和运行时可用的 vCPU 数量会影响解压缩容器映像所需要的时间。在 Fargate 上，您可以增加任务的大小或利用性能更高的 zstd 压缩算法来缩短解压缩所需要的时间。有关更多信息，请参阅 GitHub 上的 ztsd。有关如何实现 Fargate 的映像的信息，请参阅使用 zstd 压缩容器映像缩短 AWS Fargate 启动时间。

延迟加载容器映像

对于大型容器映像（> 250mb），最好延迟加载容器映像，而不是下载所有容器映像。在 Fargate 上，您可以使用 Seekable OCI（SOCI）从容器映像注册表中延迟加载容器映像。有关更多信息，请参阅 GitHub 上的 soci-snapshotter 和使用 Seekable OCI（SOCI）延迟加载容器映像。

适用于 Amazon ECS 的 Fargate Windows 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。在运行任务或创建维护多个相同任务的服务时，您可以选择平台版本。

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，将会发布新的平台版本修订版。Fargate 平台版本通过进行新的平台版本修订来更新。每个任务在其生命周期内都在一个平台版本修订版上运行。如果您想使用最新的平台版本修订版，则必须启动新任务。在 Fargate 上运行的新任务始终在平台版本的最新修订版上运行，从而确保了任务始终在安全的、经过修补的基础设施上启动。

如果发现影响现有平台版本的安全问题，AWS 将为该平台版本创建新的补丁修订版，并停止在有漏洞的修订版上运行的任务。在某些情况下，您可能会收到计划停止您在 Fargate 上的任务的通知。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate 任务维护常见问题。

平台版本注意事项

指定平台版本时请考虑以下事项：

- 指定平台版本时，您可以使用特定版本号，例如 `1.0.0` 或 `LATEST`。

  选择最新平台版本时，将使用 `1.0.0` 平台。
- 新任务始终在平台版本的最新修订版上运行，从而确保了任务始终在安全的、经过修补的基础设施上启动。
- 必须从特定版本的 Windows Server 创建 Microsoft Windows Server 容器映像。运行任务或创建与 Windows Server 容器映像相匹配的服务时，必须在 `platformFamily` 中选择相同版本的 Windows Server。此外，您可以在任务定义中提供匹配的 `operatingSystemFamily`，以防止任务在错误的 Windows 版本上运行。有关更多信息，请参阅 Microsoft Learn 网站上的将容器主机版本与容器映像版本相匹配。
- Fargate 上用于 Linux 容器和 Windows 容器的平台版本号是独立的。例如，Fargate 上的 Windows 容器的平台版本 `1.0.0` 中使用的行为、功能和软件无法与 Fargate 上的 Linux 容器的平台版本 `1.0.0` 相比。

下面是可用于 Windows 容器的平台版本。

1.0.0

以下是平台版本的更新日志 `1.0.0`。

- 初次发布，旨在支持以下 Microsoft Windows Server 操作系统：
  - Windows Server 2019 Full
  - Windows Server 2019 Core
  - Windows Server 2022 Full
  - Windows Server 2022 Core

适用于 Amazon ECS 的 Fargate 上的 Windows 容器注意事项

以下是在 AWS Fargate 上运行 Windows 容器时需要了解的区别和注意事项。

如果您需要在 Linux 和 Windows 容器上运行任务，则需要为每个操作系统创建单独的任务定义。

AWS 负责操作系统许可证管理，因此您不需要任何额外的 Microsoft Windows Server 许可证。

AWS Fargate 上的 Windows 容器支持以下操作系统：

- Windows Server 2019 Full

- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

AWS Fargate 上的 Windows 容器支持 awslog 驱动程序。有关更多信息，请参阅 the section called “将日志发送到 CloudWatch ”。

Fargate 上的 Windows 容器上不支持以下功能：

- 组托管服务账户（gMSA）
- Amazon FSx
- ENI 中继
- 针对任务的 App Mesh 服务和代理集成
- Firelens 日志路由器集成用于任务
- EFS 卷
- 以下任务定义参数：
  - `maxSwap`
  - `swappiness`
  - `environmentFiles`
- Fargate Spot 容量提供程序
- 映像卷

  Dockerfile `volume` 选项被忽略。改为在任务定义中指定绑定挂载。有关更多信息，请参阅 将绑定挂载与 Amazon ECS 结合使用。

Fargate 上的 Windows 容器的 Amazon ECS 容器映像拉取行为

Fargate Windows 缓存 Microsoft 提供的最近一个月和上个月的服务器核心基础映像。这些映像与每个补丁周二更新的 KB/Build 版本号补丁相匹配。例如，Microsoft 在 2024 年 4 月 9 日发布了适用于 Windows Server 2019 的 KB5036896（17763.5696）。上个月 2024 年 3 月 12 日的 KB 是 KB5035849（17763.5576）。因此，对于平台 `WINDOWS_SERVER_2019_CORE` 和 `WINDOWS_SERVER_2019_FULL`，已缓存以下容器映像：

- `mcr.microsoft.com/windows/servercore:ltsc2019`

- mcr.microsoft.com/windows/servercore:10.0.17763.5696
- mcr.microsoft.com/windows/servercore:10.0.17763.5576

此外，Microsoft 在 2024 年 4 月 9 日发布了适用于 Windows Server 2022 的 KB5036909（20348.2402）。上个月 2024 年 3 月 12 日的 KB 是 KB5035857（20348.2340）。因此，对于平台 WINDOWS_SERVER_2022_CORE 和 WINDOWS_SERVER_2022_FULL，已缓存以下容器映像：

- mcr.microsoft.com/windows/servercore:ltsc2022
- mcr.microsoft.com/windows/servercore:10.0.20348.2402
- mcr.microsoft.com/windows/servercore:10.0.20348.2340

适用于 Amazon ECS 的 Fargate 临时存储

预置后，AWS Fargate 上的 Linux 容器上托管的每个 Amazon ECS 任务都会收到绑定挂载的以下短暂存储。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。AWS Fargate 上的 Windows 容器不支持此选项。

Fargate Linux 容器平台版本

版本 1.4.0 或更高版本

预设情况下，使用平台版本 1.4.0 或更高版本托管在 Fargate 上的 Amazon ECS 任务获得至少 20GiB 的短暂存储。临时存储总量可以增加，最多可达 200GiB。您可以通过在任务定义中指定 ephemeralStorage 参数执行此操作。

任务的拉出、压缩和未压缩容器映像存储在临时存储中。要确定任务必须使用的临时存储总量，必须从分配的任务临时存储总量中减去容器映像使用的存储量。

对于使用平台版本 1.4.0 或更高版本且在 2020 年 5 月 28 日或之后启动的任务，将使用 AES-256 加密算法对短暂存储进行加密。此算法使用由 AWS 所有的加密密钥，您也可以使用自己的客户自主管理型密钥。有关更多信息，请参阅 Customer managed keys for AWS Fargate ephemeral storage。

对于使用平台版本 1.4.0 或更高版本且在 2022 年 11 月 18 日或之后启动的任务，将通过任务元数据端点报告短暂存储使用情况。任务中的应用程序可以查询任务元数据端点版本 4 以获取其短暂存储预留大小和已用量。

此外，如果您启用 Container Insights，短暂存储预留大小和已用量将发送到 Amazon CloudWatch Container Insights。

> **Note**
>
> Fargate 可保留磁盘空间。该磁盘空间仅由 Fargate 使用。您无需为此付费。它没有显示在这些指标中。但是，您可以在 `df` 等其他工具中看到这种额外的存储空间。

版本 1.3.0 或更早版本

对于使用平台版本 `1.3.0` 或更早版本的 Fargate 任务上的 Amazon ECS，每个任务都会收到以下临时存储。

- 10 GB 的 Docker 层存储

  > **Note**
  >
  > 此数量包括压缩和未压缩的容器映像伪影。

- 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。

Fargate Windows 容器平台版本

版本 1.0.0 或更高版本

预设情况下，使用平台版本 `1.0.0` 或更高版本托管在 Fargate 上的 Amazon ECS 任务获得至少 20GiB 的短暂存储。临时存储总量可以增加，最多可达 200GiB。您可以通过在任务定义中指定 `ephemeralStorage` 参数执行此操作。

任务的拉出、压缩和未压缩容器映像存储在临时存储中。要确定任务必须使用的临时存储总量，必须从分配的任务临时存储总量中减去容器映像使用的存储量。

有关更多信息，请参阅 将绑定挂载与 Amazon ECS 结合使用。

用于 AWS Fargate 临时存储的客户自主管理型密钥

AWS Fargate 支持使用客户自主管理型密钥来加密存储在临时存储中的 Amazon ECS 任务数据，以帮助监管敏感型客户满足其内部安全政策的需要。客户不仅可以继续享受 Fargate 的无服务器优势，同时

还可让合规审计人员更好地了解自行管理的存储加密。尽管 Fargate 默认使用由 Fargate 托管的临时存储加密，但客户在加密财务或健康相关信息等敏感数据时也可以使用自行管理的密钥。

您可以将自己的密钥导入 AWS KMS 或在 AWS KMS 中创建密钥。这些自行管理的密钥存储在 AWS KMS 中并执行标准的 AWS KMS 生命周期操作，例如轮换、禁用和删除等。您可以利用 CloudTrail 日志来审计密钥访问和使用情况。

默认情况下，每个 KMS 密钥支持 5 万个授权。Fargate 为每个客户自主管理型密钥任务使用单个 AWS KMS 授权，因此每个密钥最多支持 5 万个并发任务。如果需要增加此上限，您可以申请增加限额，但需要逐一具体审批。

Fargate 不会因使用客户自主管理型密钥而收取额外的费用。您只需按标准价格为用于存储和 API 请求的 AWS KMS 密钥付费。

主题

- 为 Fargate 临时存储创建加密密钥
- 管理 Fargate 临时存储的 AWS KMS 密钥

为 Fargate 临时存储创建加密密钥

> **Note**
>
> 使用客户自主管理型密钥加密 Fargate 临时存储的功能不适用于 Windows 任务集群。
> 使用客户自主管理型密钥加密 Fargate 临时存储的功能不适用于早于 `1.4.0` 的 `platformVersions`。
> Fargate 在临时存储中预留了仅供 Fargate 使用的空间，您无需为此空间付费。具体分配可能与非客户自主管理密钥任务不同，但总空间保持不变。您可以通过 `df` 等工具查看此更改。

要在 AWS KMS 中创建客户自主管理型密钥（CMK）来加密 Fargate 临时存储，请执行以下步骤。

1. 导航到 https://console.aws.amazon.com/kms。
2. 按照《AWS Key Management Service 开发人员指南》中 Creating Keyshttps://docs.aws.amazon.com/kms/latest/developerguide/overview.html 部分的说明操作。
3. 创建 AWS KMS 密钥时，请务必在密钥政策中提供 Fargate 服务相关 AWS KMS 操作权限。要将客户自主管理型密钥用于 Amazon ECS 集群资源，策略中必须允许以下 API 操作。

- `kms:GenerateDataKeyWithoutPlainText` – 调用 `GenerateDataKeyWithoutPlainText` 以利用提供的 AWS KMS 密钥生成加密的数据密钥。
- `kms:CreateGrant` – 向客户自主管理型密钥添加授权。这些授权会控制对指定 AWS KMS 密钥的访问权限，从而允许访问 Amazon ECS Fargate 所需的授权操作。有关使用授权的更多信息，请参阅https://docs.aws.amazon.com/kms/latest/developerguide/overview.html《AWS Key Management Service 开发人员指南》。这将允许 Amazon ECS Fargate 执行以下操作：
  - 调用 `Decrypt` 以便 AWS KMS 获取加密密钥，以便解密临时存储数据。
  - 设置停用主体，以允许服务 `RetireGrant`。
- `kms:DescribeKey` – 提供客户自主管理型密钥详细信息，以便 Amazon ECS 验证该密钥是否是对称密钥以及是否已经启用。

以下示例演示了一个将应用到目标加密密钥的 AWS KMS 密钥政策。要使用示例策略语句，请将 *`user input placeholders`* 替换为您自己的信息。与往常一样，只配置您需要的权限。

```
{
      "Sid": "Allow generate data key access for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow grant creation permission for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
```

```
        "kms:CreateGrant"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        },
       "ForAllValues:StringEquals": {
          "kms:GrantOperations": [
             "Decrypt"
          ]
       }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow describe key permission for cluster operator - CreateCluster
 and UpdateCluster.",
      "Effect": "Allow",
      "Principal": { "AWS":"arn:aws:iam::customerAccountId:role/
ClusterOperatorRole" },
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
```

Fargate 任务使用 aws:ecs:clusterAccount 和 aws:ecs:clusterName 加密上下文密钥来执行密钥的加密操作。客户应添加这些权限以限制对特定账户和/或集群的访问。

有关更多信息，请参阅 AWS KMS 开发人员指南中的加密内容。

创建或更新集群时，您可以选择使用条件键 fargateEphemeralStorageKmsKeyId。借助此条件减，客户可以更精细地控制 IAM 策略。对 fargateEphemeralStorageKmsKeyId 配置的更新仅对新的服务部署生效。

以下示例将允许客户仅向一组特定的已批准 AWS KMS 密钥授予权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ecs:fargate-ephemeral-storage-kms-key": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
      }
    }
  ]
}
```

下一个示例将拒绝删除已与集群关联的 AWS KMS 密钥的尝试。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
    ],
    "Resource": "*",
    "Condition": {
      "Null": {
        "ecs:fargate-ephemeral-storage-kms-key": "true"
      }
    }
  }
}
```

客户可以使用 AWS CLI `describe-tasks`、`describe-cluster` 或 `describe-services` 命令查看其非托管式任务或服务任务是否在使用该密钥进行加密。

有关更多信息，请参阅《AWS KMS 开发人员指南》中的 Condition keys for AWS KMShttps://docs.aws.amazon.com/kms/latest/developerguide/overview.html。

AWS Management Console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 选择左侧导航栏中的集群，然后选择右上角的创建集群，或者选择一个现有的集群。对于现有集群，请选择右上角的更新集群。
3. 在工作流的加密部分下，您可以选择在托管存储和 Fargate 临时存储下选择您的 AWS KMS 密钥。您也可以在此处选择创建 AWS KMS 密钥。
4. 创建完新集群后选择创建，或在要更新现有集群时选择更新。

AWS CLI

以下是使用 AWS CLI 创建集群并配置 Fargate 临时存储的示例（请用自己的值替换###值）：

```
aws ecs create-cluster --cluster clusterName \
--configuration '{"managedStorageConfiguration":
{"fargateEphemeralStorageKmsKeyId":"arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}}'
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:012345678901:cluster/clusterName",
        "clusterName": "clusterName",
        "configuration": {
            "managedStorageConfiguration": {
                "fargateEphemeralStorageKmsKeyId": "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [],
        "capacityProviders": [],
```

```
        "defaultCapacityProviderStrategy": []
    },
    "clusterCount": 5
}
```

AWS CloudFormation

以下是使用 AWS CloudFormation 创建集群并配置 Fargate 临时存储的示例模板（请用自己的值替换*###*值）：

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: "clusterName"
      Configuration:
        ManagedStorageConfiguration:
          FargateEphemeralStorageKmsKeyId: "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

管理 Fargate 临时存储的 AWS KMS 密钥

创建或导入 AWS KMS 密钥以加密 Fargate 临时存储后，您可以像管理任何其他 AWS KMS 密钥一样对其进行管理。

自动轮换 AWS KMS 密钥

您可以启用自动密钥轮换，也可以手动轮换密钥。自动密钥轮换通过每年为密钥生成新的加密材料来轮换密钥。AWS KMS 还会保存所有先前版本的加密材料，以便您能够解密使用早期密钥版本的任何数据。在删除密钥之前，AWS KMS 不会删除任何轮换的材料。

自动密钥轮换属于可选功能，您可以随时启用或禁用。

禁用或撤销 AWS KMS 密钥

如果您在 AWS KMS 中禁用了某个客户自主管理型密钥，这不会对正在运行的任务产生任何影响，并且这些任务将在整个生命周期中继续正常运行。如果新任务使用已禁用或已撤销的密钥，则该任务将因无法访问该密钥而失败。建议您设置 CloudWatch 警报或类似机制，来确保永远不需要使用已禁用的密钥来解密已加密的数据。

删除 AWS KMS 密钥

删除密钥应始终是最后的选择，并且仅在您确定永远不会再需要已删除的密钥时删除密钥。尝试使用已删除密钥的新任务将失败，因为这些任务将无法访问该密钥。AWS KMS 建议您禁用密钥，而不是将其删除。如果您觉得确有必要删除密钥，我们建议您首先将其禁用，然后设置 CloudWatch 警报以确保不需要该密钥。如果您确实删除了某个密钥，AWS KMS 将至少允许您在七天内改变主意。

审计 AWS KMS 密钥访问情况

您可以使用 CloudTrail 日志来审计对 AWS KMS 密钥的访问情况。您可以检查 AWS KMS 操作 `CreateGrant`、`GenerateDataKeyWithoutPlaintext` 和 `Decrypt`。这些操作还将显示 `aws:ecs:clusterAccount` 和 `aws:ecs:clusterName`，以作为 CloudTrail 中记录的 `EncryptionContext` 的一部分。

以下是 `GenerateDataKeyWithoutPlaintext`、`GenerateDataKeyWithoutPlaintext (DryRun)`、`CreateGrant`、`CreateGrant (DryRun)` 和 `RetireGrant` 的示例 CloudTrail 事件（请用您自己的值替换*###值*）。

GenerateDataKeyWithoutPlaintext


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 64,
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ebs:id": "vol-xxxxxxx",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
```

```
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

GenerateDataKeyWithoutPlaintext (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "dryRun": true,
        "numberOfBytes": 64,
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ecs:clusterName": "cluster-name"
        }
```

```
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
```

```
                "aws:ecs:clusterAccount": "account-id",
                "aws:ebs:id": "vol-xxxx",
                "aws:ecs:clusterName": "cluster-name"
            }
        },
        "retiringPrincipal": "ec2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "dryRun": true,
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ecs:clusterName": "cluster-name"
            }
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

RetireGrant

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2024-04-20T18:37:38Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "additionalEventData": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

Amazon ECS 上的 AWS Fargate 任务维护常见问题

什么是 Fargate 任务维护和停用？

AWS 负责维护 AWS Fargate 的底层基础设施。AWS 确定何时需要将平台版本修订版替换为新的基础设施修订版。这称为任务停用。当平台版本版本修订停用时，AWS 会发送任务停用通知。我们会定期更新支持的平台版本以引入新版本，其中包含对 Fargate 运行时软件，以及底层依赖项（例如操作系统和容器运行时）的更新。提供较新修订版后，我们会停用旧版本，以确保所有客户工作负载均在最新修订版的 Fargate 平台版本上运行。当停用修订版时，系统会停用在该修订版上运行的所有任务。

Amazon ECS 任务可以分为服务任务和独立任务。服务任务作为服务的一部分进行部署，并由 Amazon ECS 计划控制。有关更多信息，请参阅 Amazon ECS 服务。独立任务是由 Amazon ECS RunTask API 直接启动或通过外部调度器启动的任务，例如计划任务（由 Amazon EventBridge 启动）、AWS Batch 或 AWS Step Functions。

对于服务任务，除非您想在 AWS 执行前替换这些任务，否则无需执行任何操作。当 Amazon ECS 计划程序停止任务时，其使用最小正常百分比并启动一个新任务，以尝试保持服务所需的任务数。默认情况下，服务的最小正常百分比为 100%，因此在停止任务之前应先启动新任务。在扩缩服务、部署配置更改或部署任务定义修订版时，通常以相同的方式替换服务任务。为了准备任务停用过程，我们建议通过模拟此场景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。

对于独立任务停用，AWS 将在任务停用日期当天或之后停止该任务。我们不会在任务停止时启动替换任务。如果您需要继续运行这些任务，则需要在通知中指示的时间之前停止正在运行的任务并启动替换任务。因此，我们建议客户监控独立任务的状态，并在需要时实施逻辑来替换已停止的任务。

在任何情景中停止任务时，您都可以运行 `describe-tasks`。响应中的 `stoppedReason` 是 `ECS is performing maintenance on the underlying infrastructure hosting the task`。

当有新的平台版本修订版并需要替换为新修订版时，任务维护适用。如果底层 Fargate 主机出现问题，则 Amazon ECS 会更换主机而不发出任务停用通知。

任务停用通知中包含哪些内容？

任务停用通知通过 AWS Health Dashboard，以及通过电子邮件发送到注册的电子邮件地址，其中包含以下信息：

- 任务停用日期 - 任务在此日期或之后停止。

- 对于独立任务，任务的 ID。
- 对于服务任务，运行服务的集群 ID 和服务 ID。
- 您需要采取的后续步骤。

通常，我们会针对每个 AWS 区域 中的服务和独立任务各发送一条通知。但是，在某些情况下，每种任务类型可能会收到多个事件。例如，有太多要停用的任务，将超出我们通知机制中的限制时。

通过以下方式，您可以确定计划停用的任务：

- 这些区域有：AWS Health Dashboard

  您可以通过 Amazon EventBridge 将 AWS Health 通知发送到存档存储器（例如 Amazon Simple Storage Service）、执行自动操作（例如运行 AWS Lambda 函数）或其他通知系统（例如 Amazon Simple Notification Service）。有关更多信息，请参阅使用 Amazon EventBridge 监控 AWS Health 事件。有关向 Amazon Chime、Slack 或 Microsoft Teams 发送通知的示例配置，请参阅 GitHub 上的 AWS Health Aware 存储库。

  以下是一个示例 EventBridge 事件。

```
{
    "version": "0",
    "id": "3c268027-f43c-0171-7425-1d799EXAMPLE",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-08-16T23:18:51Z",
    "region": "us-east-1",
    "resources": [
        "cluster/service",
        "cluster/service"
    ],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/ECS/
AWS_ECS_TASK_PATCHING_RETIREMENT/AWS_ECS_TASK_PATCHING_RETIREMENT_test1",
        "service": "ECS",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
        "communicationId":
 "7988399e2e6fb0b905ddc88e0e2de1fd17e4c9fa60349577446d95a18EXAMPLE",
        "lastUpdatedTime": "Wed, 16 Aug 2023 23:18:52 GMT",
        "eventRegion": "us-east-1",
        "eventTypeCode": "AWS_ECS_TASK_PATCHING_RETIREMENT",
```

```
        "eventTypeCategory": "scheduledChange",
        "startTime": "Wed, 16 Aug 2023 23:18:51 GMT",
        "endTime": "Fri, 18 Aug 2023 23:18:51 GMT",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "\\nA software update has been deployed to
 Fargate which includes CVE patches or other critical patches. No action is required
 on your part. All new tasks launched automatically uses the latest software
 version. For existing tasks, your tasks need to be restarted in order for these
 updates to apply. Your tasks running as part of the following ECS Services will
 be automatically updated beginning Wed, 16 Aug 2023 23:18:51 GMT.\\n\\nAfter Wed,
 16 Aug 2023 23:18:51 GMT, the ECS scheduler will gradually replace these tasks,
 respecting the deployment settings for your service. Typically, services should
 see little to no interruption during the update and no action is required. When AWS
 stops tasks, AWS uses the minimum healthy percent (1) and launches a new task in
 an attempt to maintain the desired count for the service. By default, the minimum
 healthy percent of a service is 100 percent, so a new task is started first before
 a task is stopped. Service tasks are routinely replaced in the same way when
 you scale the service or deploy configuration changes or deploy task definition
 revisions. If you would like to control the timing of this restart you can update
 the service before Wed, 16 Aug 2023 23:18:51 GMT, by running the update-service
 command from the ECS command-line interface specifying force-new-deployment for
 services using Rolling update deployment type. For example:\\n\\n$ aws ecs update-
service -service service_name \\\n--cluster cluster_name -force-new-deployment\
\n\\nFor services using Blue/Green deployment type with AWS CodeDeploy:\\nPlease
 refer to create-deployment document (2) and create new deployment using same task
 definition revision.\\n\\nFor further details on ECS deployment types, please
 refer to ECS Deployment Developer Guide (1).\\nFor further details on Fargate's
 update process, please refer to the AWS Fargate User Guide (3).\\nIf you have
 any questions or concerns, please contact AWS Support (4).\\n\\n(1) https://
docs.aws.amazon.com/AmazonECS/latest/developerguide/deployment-types.html\\n(2)
 https://docs.aws.amazon.com/cli/latest/reference/deploy/create-deployment.html\\n(3)
 https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-maintenance.html\\n(4)
 https://aws.amazon.com/support\\n\\nA list of your affected resources(s) can be
 found in the 'Affected resources' tab in the 'Cluster/ Service' format in the AWS
 Health Dashboard. \\n\\n"
            }
        ],
        "affectedEntities": [
            {
                "entityValue": "cluster/service"
            },
            {
```

```
                "entityValue": "cluster/service"
            }
        ]
    }
}
```

- Email

  将向注册的电子邮件发送一封电子邮件以获取 AWS 账户 ID。

我可以更改任务停用等待时间吗？

您可以对 Fargate 开始停用任务的时间进行配置。对于需要立即应用更新的工作负载，请选择即时设置（0）。当您需要更多控制时，例如，当任务只能在特定时间段内停止时，请配置 7 天（7）或 14 天（14）选项。

建议您选择较短的等待时间，以便更快获得较新的平台版本修订版。

通过以根用户或管理员用户身份运行 `put-account-setting-default` 或 `put-account-setting` 配置等待期。将 `fargateTaskRetirementWaitPeriod` 选项用于设置为以下值之一的 `name` 和 `value` 选项：

- `0` - AWS 发送通知，并立即开始停用受影响的任务。
- `7` - AWS 发送通知，等待 7 个日历日后才开始停用受影响的任务。
- `14` - AWS 发送通知，等待 14 个日历日后才开始停用受影响的任务。

默认值为 7 天。

有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 put-account-setting-default 和 put-account-setting。

有关更多信息，请参阅 AWS Fargate 任务停用等待时间。

我可以通过其他 AWS 服务获取任务停用通知吗？

AWS 向 AWS Health Dashboard 和 AWS 账户 上的主要电子邮件联系人发送任务停用通知。AWS Health Dashboard 提供了与其他 AWS 服务（包括 EventBridge）的许多集成。您可以使用 EventBridge 自动显示通知（例如，将消息转发到 ChatOps 工具）。有关更多信息，请参阅 Solution overview: Capturing task retirement notifications。

我可以在任务停用计划后更改任务停用吗？

不可以。该计划基于任务停用等待时间，默认值为 7 天。如果您需要更多时间，则可以选择将等待期配置为 14 天。有关更多信息，请参阅 我可以更改任务停用等待时间吗？。此配置的更改适用于将来计划的停用。目前已计划的停用不受影响。如有任何疑问，请联系 AWS Support。

我可以控制任务替换的时间吗？

对于使用滚动部署的服务，您可以在停用开始时间之前使用带 `force-deployment` 选项的 `update-service` 来更新服务。

以下 `update-service` 示例使用 `force-deployment` 选项。

```
aws ecs update-service –-service service_name \
    --cluster cluster_name \
     --force-new-deployment
```

对于使用蓝/绿部署的服务，您需要在 AWS CodeDeploy 中创建新部署。有关如何创建部署的信息，请参阅《AWS Command Line Interface Reference》中的 create-deployment。

Amazon ECS 如何处理属于服务一部分的任务？

当 Fargate 停用期开始时，Amazon ECS 会逐渐替换服务中受影响的任务。当 Amazon ECS 停止某项任务时，其使用服务的最小正常百分比并启动一个新任务，以保持服务所需的任务数。由于默认的最小正常百分比为 100，因此新任务会在任务停止之前启动。在扩缩服务、部署配置更改或部署任务定义修订版时，通常以相同的方式替换服务任务。有关最小正常百分比的更多信息，请参阅部署配置。

Amazon ECS 可以自动处理独立任务吗？

不可以。AWS 无法为通过 `RunTask`、计划任务（例如通过 EventBridge 调度器）、AWS Batch 或 AWS Step Functions 启动的独立任务创建替换任务。Amazon ECS 仅管理属于服务一部分的任务。

Amazon ECS 在 AWS Fargate 上的支持区域

您可以使用下表来验证 AWS Fargate 上的 Linux 容器和 AWS Fargate 上的 Windows 容器的区域支持。

AWS Fargate 上的 Linux 容器

AWS Fargate 上的 Amazon ECS Linux 容器在以下 AWS 区域 中受支持。根据需要记录支持的可用区 ID。

区域名称	区域
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（北加利福尼亚）	us-west-1 (仅限 usw1-az1 & usw1-az3)
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（孟买）	ap-south-1
亚太地区（东京)	ap-northeast-1 (仅限 apne1-az1 、apne1-az2 &apne1-az4)
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
亚太地区（海得拉巴）	ap-south-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（墨尔本）	ap-southeast-4
加拿大（中部）	ca-central-1
加拿大西部（卡尔加里）	ca-west-1

区域名称	区域
中国（北京）	cn-north-1 (仅限 cnn1-az1 & cnn1-az2)
中国（宁夏）	cn-northwest-1
欧洲（法兰克福）	eu-central-1
欧洲（苏黎世）	eu-central-2
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3
欧洲（米兰）	eu-south-1
欧洲（西班牙）	eu-south-2
欧洲地区（斯德哥尔摩）	eu-north-1
南美洲（圣保罗）	sa-east-1
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
AWS GovCloud（美国东部）	us-gov-east-1
AWS GovCloud（美国西部）	us-gov-west-1

AWS Fargate 上的 Windows 容器

AWS Fargate 上的 Amazon ECS Windows 容器在以下 AWS 区域 中受支持。根据需要记录支持的可用区 ID。

区域名称	区域
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1（仅限 use1-az1、use1-az2、use1-az4、use1-az5 和 use1-az6）
美国西部（加利福尼亚北部）	us-west-1 (仅限 usw1-az1 & usw1-az3）
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（墨尔本）	ap-southeast-4
亚太地区（东京)	ap-northeast-1 (仅限 apne1-az1 、apne1-az2 &apne1-az4 ）
加拿大（中部）	ca-central-1 (仅限 cac1-az1 & cac1-az2）
加拿大西部（卡尔加里）	ca-west-1
中国（北京）	cn-north-1 (仅限 cnn1-az1 & cnn1-az2)
中国（宁夏）	cn-northwest-1
欧洲（法兰克福）	eu-central-1

区域名称	区域
欧洲（苏黎世）	eu-central-2
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3
欧洲（米兰）	eu-south-1
欧洲（西班牙）	eu-south-2
欧洲地区（斯德哥尔摩）	eu-north-1
南美洲（圣保罗）	sa-east-1
以色列（特拉维夫）	il-central-1
中东（阿联酋）	me-central-1
中东（巴林）	me-south-1

为 Amazon ECS 构建您的解决方案

在使用 Amazon ECS 之前，您需要对容量、联网、账户设置和日志记录进行决策，以便正确配置您的 Amazon ECS 资源。

容量

容量是容器运行的基础设施。选项如下：

- Amazon EC2 实例
- 无服务器（AWS Fargate (Fargate)）
- 本地虚拟机（VM）或服务器

在您创建集群时指定基础设施。您还可以在注册任务定义时指定基础设施类型。任务定义将基础设施称为“启动类型”。当您运行独立任务或部署服务时，也可以使用启动类型。有关启动类型选项的更多信息，请参阅Amazon ECS 启动类型。

联网

AWS 资源是在子网中创建的。当您使用 EC2 实例时，Amazon ECS 会在您在创建集群时指定的子网中启动实例。您的任务在实例子网中运行。对于 Fargate 或本地虚拟机，您可以在运行任务或创建服务时指定子网。

根据您的应用程序，子网可以是私有子网或公有子网，并且子网可以位于任何以下 AWS 资源中：

- 可用区
- Local Zones
- Wavelength 区域
- AWS 区域
- AWS Outposts

有关更多信息，请参阅Amazon ECS 应用程序位于共享子网、本地区和 Wavelength 区中或AWS Outposts 上的 Amazon Elastic Container Service。

您可以使用以下方法之一将应用程序连接到互联网：

- 具有互联网网关的公有子网

  当您的公共应用程序需要大量带宽或最小延迟时，请使用公有子网。适用场景包括视频流和游戏服务。

- 具有 NAT 网关的私有子网

  如果要保护容器免于外部直接访问，则请使用私有子网。适用场景包括支付处理系统或存储用户数据和密码的容器。

功能访问

您可以使用 Amazon ECS 账户设置来访问以下功能：

- Container Insights

  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。

- `awsvpc` 中继

  对于某些 EC2 实例类型，您可以在新启动的容器实例上使用其他网络接口（ENI）。

- 标记授权

  用户必须具有创建资源的操作的权限，例如 `ecsCreateCluster`。如果在资源创建操作中指定了标签，则 AWS 会对 `ecs:TagResource` 操作执行额外的授权，以验证用户或角色是否具备创建标签的权限。

- Fargate FIPS-140 合规性

  Fargate 支持美国联邦信息处理标准（FIPS-140），其中规定了对保护敏感信息的加密模块的安全要求。它是美国和加拿大政府的现行标准，适用于需要符合《联邦信息安全管理法》（FISMA）或联邦风险和授权管理计划（FedRAMP）的系统。

- Fargate 任务停用时间变更

  您可以配置对停用 Fargate 任务进行修补之前的等待时间。

- 双堆栈 VPC

  允许任务通过 IPv4、IPv6 或两者进行通信。

- Amazon Resource Name (ARN) 格式

某些功能（例如标记授权）需要新的 Amazon 资源名称（ARN）格式。

有关更多信息，请参阅 通过账户设置访问 Amazon ECS 功能。

IAM 角色

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。在 Amazon ECS 中，您可以创建角色来授予对 Amazon ECS 资源（例如容器或服务）的权限。

有些 Amazon ECS 功能需要角色。有关更多信息，请参阅 适用于 Amazon ECS 的 IAM 角色。

日志记录

日志记录和监控是维护 Amazon ECS 工作负载的可靠性、可用性和性能的重要方面。以下选项可用：

- Amazon CloudWatch 日志 – 将日志路由到 Amazon CloudWatch
- FireLens for Amazon ECS – 将日志路由到 AWS 服务或 AWS Partner Network 目的地，以进行日志存储和分析。AWS Partner Network 是一个由合作伙伴组成的全球社区，它利用计划、专业知识和资源来构建、营销和销售客户产品。

Amazon ECS 启动类型

任务定义启动类型定义了任务可以运行的容量，例如 AWS Fargate。

选择启动类型后，Amazon ECS 会验证您配置的任务定义参数是否适用于启动类型。

Fargate

Fargate 是一款无服务器、按实际使用量付费的计算引擎，让您无需管理服务器即可专注于构建应用程序。选择 Fargate 时，您无需管理 EC2 基础设施。您只需构建容器映像，并定义要在哪个集群上运行应用程序。Fargate 与以下 AWS 服务进行本机集成，包括：

- Amazon VPC
- Auto Scaling
- Elastic Load Balancing

- IAM
- Secrets Manager

与 EC2 相比，您对 Fargate 拥有更多控制权，因为您可以选择应用程序所需的确切 CPU 和内存。Fargate 可以横向扩展容量，因此您无需担心流量激增。这意味着 Fargate 的运营工作量更少。

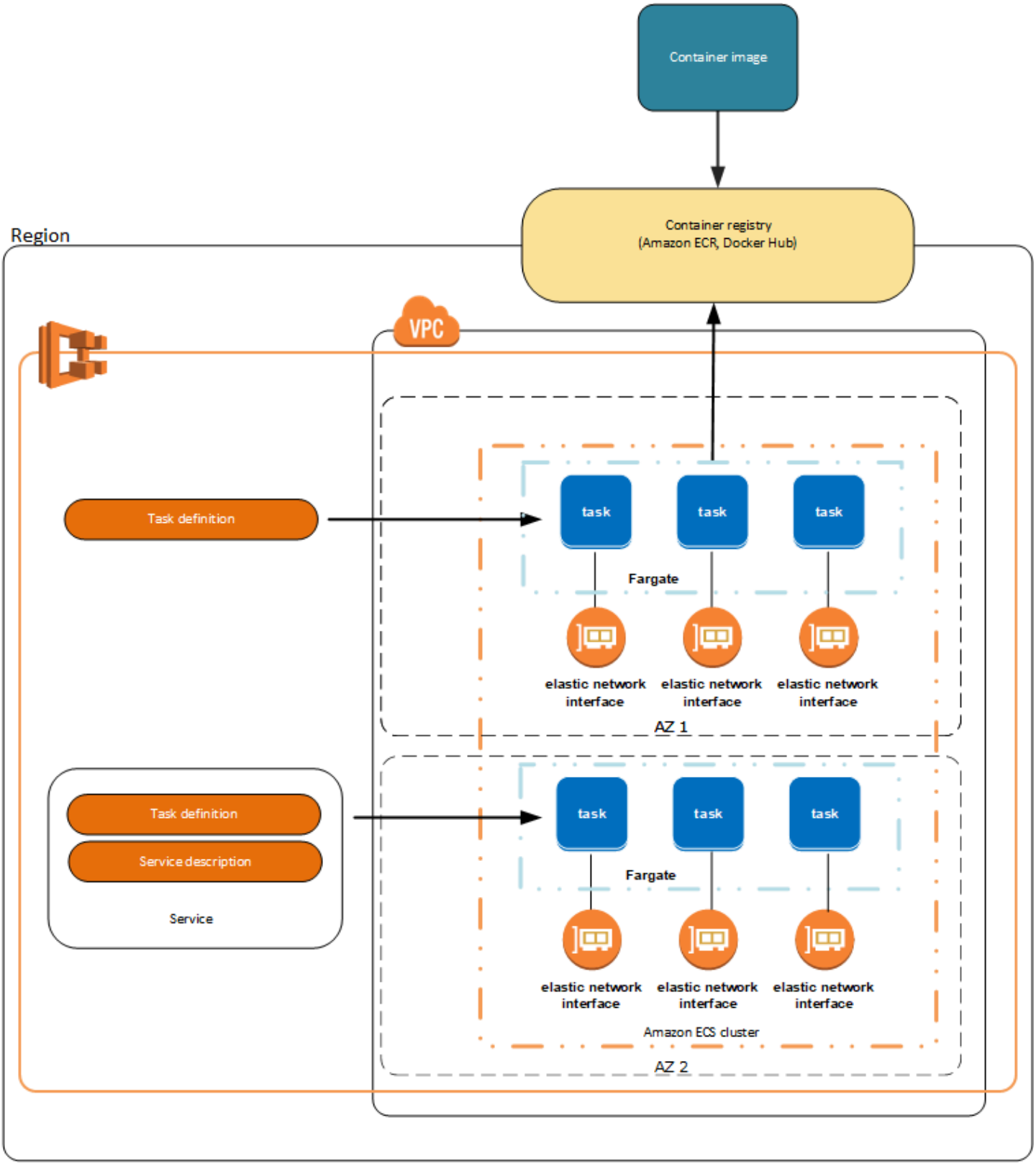
Fargate 符合合规计划的标准，包括 PCI、FIPS 140-2、FedRAMP 和 HIPAA。有关更多信息，请参阅合规性计划范围内的 AWS 服务。

Fargate 适用于以下工作负载：

- 需要低运营开销的大型工作负载
- 偶尔会突增的小型工作负载
- 小工作负载
- 批处理工作负载

有关支持 Fargate 的区域的信息，请参阅 the section called “AWS Fargate 区域”。

下图演示了一般架构。



有关 Fargate 上的Amazon ECS 更多信息，请参阅 适用于 Amazon ECS 的 AWS Fargate。

EC2

EC2 启动类型适用于必须进行价格优化的大型工作负载。

当考虑如何使用 EC2 启动类型对任务定义和服务建模时，我们建议您考虑哪些流程必须一起运行以及您将如何扩展每个组件。

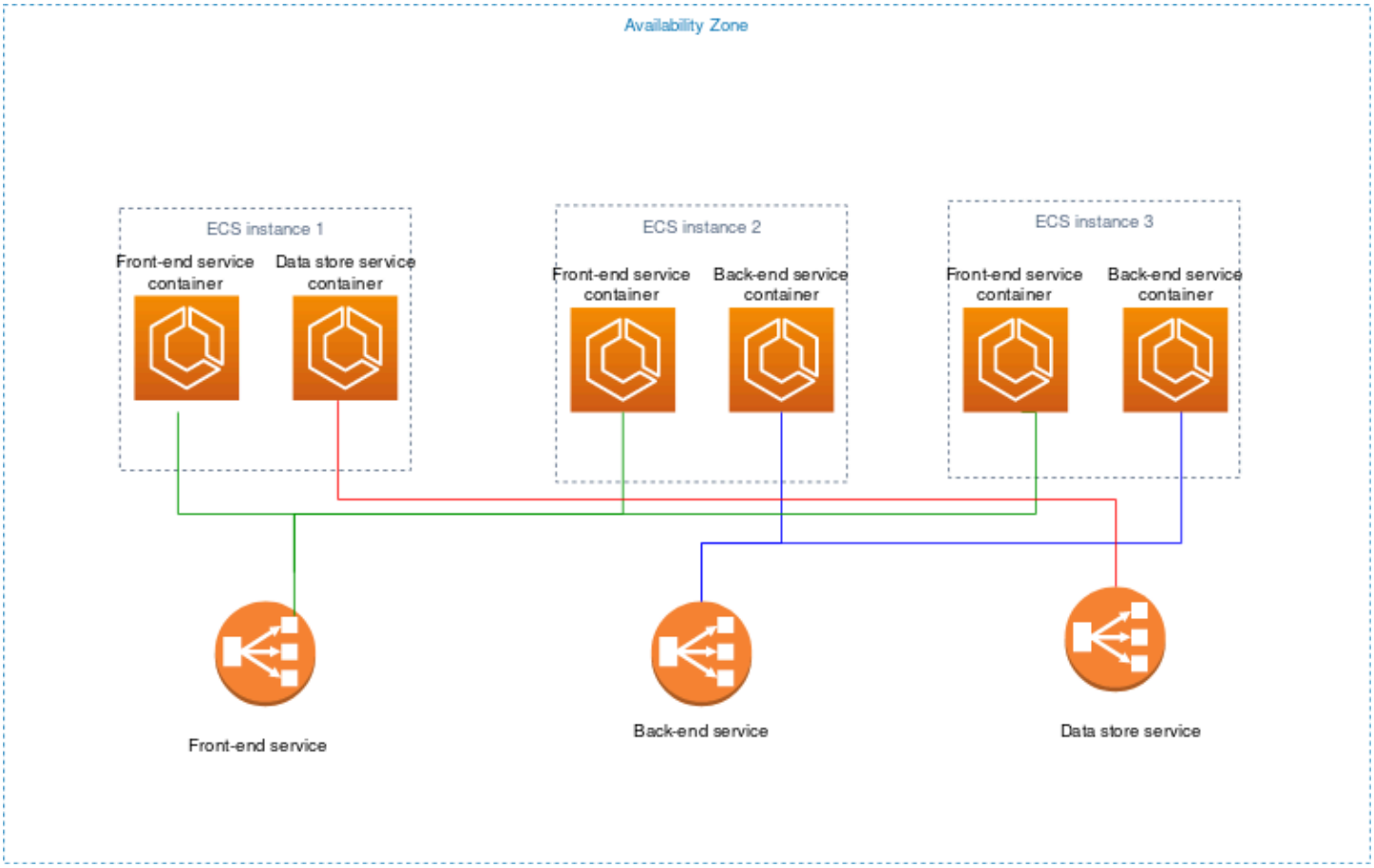
例如，假设某个应用程序包含以下组件：

- 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

对于此示例，创建将用于共同目的的容器分组的任务定义。将不同的组件分离为多个单独的任务定义。以下示例集群具有运行三个前端服务容器、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，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，请参阅使用控制台创建 Amazon ECS 服务。在您的服务中，您可以将容器与 Elastic Load Balancing 负载平衡器关联。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。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，您可以更新服务以在任务中部署较新版本的容器。有关更多信息，请参阅 使用控制台更新 Amazon ECS 服务。



外部

外部启动类型用于在您注册到 Amazon ECS 群集并远程管理的内部部署服务器或虚拟机 (VM) 上运行容器化应用程序。有关更多信息，请参阅 外部启动类型的 Amazon ECS 集群。

Amazon ECS 应用程序位于共享子网、本地区和 Wavelength 区中

Amazon ECS 支持利用 Local Zone、Wavelength 区域和 AWS Outposts 的工作负载，用于需要低延迟或本地数据处理的情况。

- 可以使用本地区作为 AWS 区域 的扩展，让您可以在多个离最终用户较近的位置放置资源。
- 您可以使用 Wavelength Zones 为 5G 设备和最终用户打造具有超低延迟的应用程序。Wavelength 可以将标准 AWS 计算和存储服务部署到电信运营商的 5G 网络边缘。
- AWS Outposts 可将本机 AWS 服务 服务、基础设施和运营模式引入几乎任何数据中心、主机托管空间或本地设施。

> **Important**
>
> 目前，Local Zones、Wavelength Zones 或 AWS Outposts 不支持 AWS Fargate 工作负载上的 Amazon ECS。

有关 Local Zones、Wavelength Zones 和 AWS Outposts 之间差异的信息，请参阅 AWS Wavelength 常见问题中的我应该如何考虑何时对需要低延迟或本地数据处理的应用程序使用 AWS Wavelength、AWS Local Zones 或 AWS Outposts。

共享子网

您可以使用 VPC 共享与同一 AWS Organizations 中的其他 AWS 账户共享子网。

您可以将共享 VPC 用于 EC2 启动类型，但需要注意以下事项：

- VPC 子网的所有者必须与参与者账户共享一个子网，然后该账户才能将其用于 Amazon EKS 资源。
- 您无法对容器实例使用 VPC 默认安全组，因为此安全组属于所有者。此外，参与者无法使用其他参与者或所有者拥有的安全组启动实例。
- 在共享子网中，参与者和拥有者分别控制各自账户中的安全组。子网拥有者可以看到参与者创建的安全组，但不能对其执行任何操作。如果子网拥有者想要删除或修改安全组，则创建安全组的参与者必须执行该操作。
- 共享 VPC 拥有者无法查看、更新或删除参与者在共享子网中创建的集群。这是对每个账户具有不同访问权限的 VPC 资源的补充。有关更多信息，请参阅 Amazon VPC 用户指南中的拥有者和参与者的责任和权限。

您可以将共享 VPC 用于 Fargate 启动类型，但需要注意以下事项：

- VPC 子网的所有者必须与参与者账户共享一个子网，然后该账户才能将其用于 Amazon EKS 资源。
- 您不能使用 VPC 的默认安全组创建服务或运行任务，因为此安全组属于其所有者。此外，参与者无法使用其他参与者或所有者拥有的安全组创建服务或运行任务。
- 在共享子网中，参与者和拥有者分别控制各自账户中的安全组。子网拥有者可以看到参与者创建的安全组，但不能对其执行任何操作。如果子网拥有者想要删除或修改安全组，则创建安全组的参与者必须执行该操作。
- 共享 VPC 拥有者无法查看、更新或删除参与者在共享子网中创建的集群。这是对每个账户具有不同访问权限的 VPC 资源的补充。有关更多信息，请参阅 Amazon VPC 用户指南中的拥有者和参与者的责任和权限。

有关 VPC 子网共享的更多信息，请参阅 Amazon VPC 用户指南中的与其他账户共享 VPC。

Local Zones

Local Zone 是在地理上靠近您的用户的 AWS 区域 的扩展。Local Zones 有自己的 Internet 连接并支持 AWS Direct Connect。在 Local Zones 中创建的资源可以通过低延迟通信服务于本地用户。有关更多信息，请参阅 AWS Local Zones。

Local Zones 由区域代码后跟一个指示位置的标识符表示（例如 `us-west-2-lax-1a`）。

要使用 Local Zone，您必须首先选择加入该区域。选择加入后，您必须在 Local Zone 中创建 Amazon VPC 和子网。

您可以启动 Amazon EC2 实例、Amazon FSx 文件服务器和应用程序负载均衡器，以用于 Amazon ECS 集群和任务。

有关更多信息，请参阅《AWS Local Zones 用户指南》中的 What is AWS Local Zones?。

Wavelength 区域

您可以使用 AWS Wavelength 为移动设备和最终用户打造具有超低延迟的应用程序。Wavelength 可以将标准 AWS 计算和存储服务部署到电信运营商的 5G 网络边缘。您可以将 Amazon Virtual Private Cloud 扩展到一个或多个 Wavelength Zones。然后，您可以使用 Amazon EC2 实例之类的 AWS 资源来运行需要超低延迟和连接到该区域中的 AWS 服务 的应用程序。

Wavelength Zone 是在其中部署 Wavelength 基础设施的运营商位置中的隔离区域。Wavelength Zone 与 AWS 区域 相关联。Wavelength 区域是区域的逻辑扩展，由区域中的控制平面管理。

Wavelength Zone 由区域代码后跟一个指示 Wavelength Zone 的标识符表示（例如 `us-east-1-wl1-bos-wlz-1`）。

要使用 Wavelength Zone，您必须选择加入此区域。选择加入后，您必须在 Wavelength Zone 中创建 Amazon VPC 和子网。然后，您可以在该区域中启动 Amazon EC2 实例，以用于 Amazon EC2 集群和任务。

有关更多信息，请参阅 AWS Wavelength 开发人员指南中的 AWS Wavelength 入门。

并非所有 AWS 区域 都支持 Wavelength Zone。有关支持 Wavelength 区域的区域的信息，请参阅 AWS Wavelength 开发人员指南中的可用 Wavelength 区域。

AWS Outposts 上的 Amazon Elastic Container Service

AWS Outposts 在本地设施中启用本机 AWS 服务、基础设施和操作模型。在 AWS Outposts 环境中，您可以使用与 AWS Cloud 中相同的 AWS API、工具和基础设施。

AWS Outposts 上的 Amazon ECS 非常适合需要靠近本地数据和应用程序运行的低延迟工作负载。

有关 AWS Outposts 的更多信息，请参阅《AWS Outposts 用户指南》。https://docs.aws.amazon.com/outposts/latest/userguide/what-is-outposts.html

注意事项

以下是在 AWS Outposts 上使用 Amazon ECS 的注意事项：

- Amazon Elastic Container Registry、AWS Identity and Access Management 和网络负载均衡器在 AWS 区域而不是 AWS Outposts 上运行。这将增加这些服务和容器之间的延迟。
- AWS Fargate 在 AWS Outposts 上不可用。

以下是 AWS Outposts 的网络连接注意事项：

- 如果您的 AWS Outposts 与其 AWS 区域之间的网络连接丢失，您的集群将继续运行。但是，在恢复连接之前，您无法创建新集群或对现有集群执行新操作。如果实例出现故障，则不会自动替换该实例。CloudWatch Logs 代理将无法更新日志和事件数据。
- 建议在 AWS Outposts 与其 AWS 区域之间提供可靠且高度可用的低延迟连接。

先决条件

以下是在 AWS Outposts 上使用 Amazon ECS 的先决条件：

- 您必须已在本地数据中心中安装并配置了 Outpost。
- Outpost 与其 AWS 区域之间必须具有可靠的网络连接。

在 AWS Outposts 上创建集群

要使用 AWS CLI 在 AWS Outposts 上创建 Amazon ECS 集群，请指定与您的 AWS Outposts 关联的安全组和子网 。

创建与您的 AWS Outposts 关联的子网。

```
aws ec2 create-subnet \
    --cidr-block 10.0.3.0/24 \
    --vpc-id vpc-xxxxxxxx \
    --outpost-arn arn:aws:outposts:us-west-2:123456789012:outpost/op-xxxxxxxxxxxxxxxx \
    --availability-zone-id usw2-az1
```

以下示例在 AWS Outposts 上创建一个 Amazon ECS 集群。

1. 创建具有 AWS Outposts 权限的角色和策略。

   `role-policy.json` 文件是包含资源效果和操作的策略文档。有关文件格式的信息，请参阅 IAM API 参考中的 PutRolePolicy

   ```
   aws iam create-role --role-name ecsRole \
       --assume-role-policy-document file://ecs-policy.json
   aws iam put-role-policy --role-name ecsRole --policy-name ecsRolePolicy \
       --policy-document file://role-policy.json
   ```

2. 在 AWS Outposts 创建具有权限的 IAM 实例配置文件。

   ```
   aws iam create-instance-profile --instance-profile-name outpost
   aws iam add-role-to-instance-profile --instance-profile-name outpost \
       --role-name ecsRole
   ```

3. 创建 VPC。

   ```
   aws ec2 create-vpc --cidr-block 10.0.0.0/16
   ```

4. 为容器实例创建安全组，为 AWS Outposts 指定适合的 CIDR 范围。（此步骤不同于 AWS Outposts。）

   ```
   aws ec2 create-security-group --group-name MyOutpostSG
   aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
       --port 22 --cidr 10.0.3.0/24
   aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
       --port 80 --cidr 10.0.3.0/24
   ```

5. 创建集群。

6. 定义 Amazon ECS 容器代理环境变量以将实例启动到上一步创建的集群中，并定义您想添加的所有标签以帮助识别集群（例如，`Outpost` 表示集群用于 Outpost）。

```
#! /bin/bash
cat << 'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_IMAGE_PULL_BEHAVIOR=prefer-cached
ECS_CONTAINER_INSTANCE_TAGS={"environment": "Outpost"}
EOF
```

> **Note**
>
> 为了避免因从区域中的 Amazon ECR 拉取容器映像而导致的延迟，请使用映像缓存。为此，每次运行任务时，请将 Amazon ECS 代理配置为默认使用实例本身的缓存映像，方法是将 ECS_IMAGE_PULL_BEHAVIOR 设置为 prefer-cached。

7. 创建容器实例，指定该实例 AWS Outposts 应运行的VPC和子网以及 AWS Outposts 上可用的实例类型。（此步骤不同于 AWS Outposts。）

   userdata.txt 文件包含用户数据，实例可使用这些数据执行常见的自动配置任务，甚至是在实例启动后运行脚本。有关 API 调用文件的信息，请参阅《Amazon EC2 用户指南》中的启动时在 Linux 实例上运行命令。

   ```
   aws ec2 run-instances --count 1 --image-id ami-xxxxxxxx --instance-type c5.large \
       --key-name aws-outpost-key --subnet-id subnet-xxxxxxxxxxxxxxxxx \
       --iam-instance-profile Name outpost --security-group-id sg-xxxxxx \
       --associate-public-ip-address --user-data file://userdata.txt
   ```

   > **Note**
   >
   > 向集群添加其他实例时，也使用此命令。集群中部署的任何容器都将置于该特定 AWS Outposts 上。

8. 注册您的任务定义。使用以下命令并将 ecs-task.json 替换为您的任务定义名称。

   ```
   aws ecs register-task-definition --cli-input-json file://ecs-task.json
   ```

9. 运行任务或创建服务。

Run the task

```
aws ecs run-task --cluster mycluster --count 1 --task-definition outpost-app:1
```

Create the service

```
aws ecs create-service --cluster mycluster --service-name outpost-service \
    --task-definition outpost-app:1 --desired-count 1
```

优化 Amazon ECS 容量和可用性

应用程序可用性对于提供无错误体验和最大限度降低应用程序延迟至关重要。可用性取决于是否拥有可访问的资源并有足够的容量来满足需求。AWS 提供了几种管理可用性的机制。对于托管在 Amazon ECS 上的应用程序，这些机制包括自动扩缩和可用区（AZ）。自动扩缩根据您定义的指标管理任务或实例的数量，而可用区则允许您将应用程序托管在孤立但地理位置接近的位置。

与任务规模一样，容量和可用性也存在您必须考虑的某些权衡。理想情况下，容量将与需求完全一致。总是有足够的容量来提供请求和处理作业，以满足服务级别目标（SLO），包括低延迟和错误率。容量永远不会太高，导致成本过高；也永远不会太低，导致高延迟和错误率。

自动扩缩是一个潜在过程。首先，必须将实时指标传输至 CloudWatch。然后，需要将它们汇总起来以进行分析，这可能需要长达几分钟的时间，具体取决于指标的粒度。CloudWatch 将这些指标与警报阈值进行比较，以确定资源短缺或过剩情况。为防止不稳定，配置警报，要求设置的阈值在警报响起前几分钟被超过。预调配新任务和终止不再需要的任务也需要时间。

由于所描述的系统存在这些潜在的延迟，因此通过过度配置来保持一定的余量很重要。这样做有助于适应短期的需求激增。这也有助于您的应用程序在不达到饱和的情况下服务额外的请求。作为一种良好的实践，您可以将扩缩目标设置在利用率的 60-80% 之间。这可以帮助您的应用程序更好地处理额外需求激增，同时仍在预调配更多容量。

建议您过度配置的另一个原因是，您可以快速响应可用区故障。AWS 建议从多个可用区提供生产工作负载。这是因为，如果一个可用区发生故障，则在剩余可用区中运行的任务仍然可以满足需求。如果应用程序在两个可用区中运行，则需要将正常任务数增加一倍。这样，您就可以在任何潜在故障期间提供即时容量。如果您的应用程序在三个可用区中运行，则建议您运行正常任务数的 1.5 倍。也就是说，对正常服务所需的每两个任务执行三个任务。

最大程度提高缩放速度

自动扩缩是一个被动的过程，需要时间才能生效。但是，有一些方法可以帮助最大限度地缩短横向扩展所需的时间。

最小化映像大小。较大的映像需要更长的时间才能从映像存储库中下载和解压缩。因此，缩小映像大小可以减少容器启动所需的时间。要缩小映像大小，您可以遵循以下具体建议：

- 如果您可以构建静态二进制文件或使用 Golang，则请 FROM 从头开始构建映像，并在生成的映像中仅包含您的二进制应用程序。
- 使用来自上游 Distro 供应商（例如 Amazon Linux 或 Ubuntu）的最小化基础映像。
- 请勿在最终映像中包含任何生成构件。使用多阶段生成可以帮助解决这个问题。
- 尽可能压缩 RUN 阶段。每个 RUN 阶段都会创建一个新的映像层，从而发起另一轮下载层的循环。具有由 && 联接的多个命令的单个 RUN 阶段的层数少于具有多个 RUN 阶段的命令。
- 如果您想在最终映像中包含机器学习推理数据等数据，则请仅包含启动和开始提供流量所需的数据。如果您在不影响服务的情况下按需从 Amazon S3 或其他存储中提取数据，则请改为将数据存储在这些位置。

保持映像相互靠近。网络延迟越高，下载映像所需的时间越长。将您的映像托管在与您的工作负载所在 AWS 区域相同的存储库中。Amazon ECR 是一个高性能映像存储库，在 Amazon ECS 所在的每个区域都可用。避免遍历互联网或使用 VPN 链接下载容器映像。将您的映像托管在同一区域可以提高整体可靠性。它缓解了不同区域出现网络连接问题和可用性问题的风险。或者，您还可以实施 Amazon ECR 跨区域复制来帮助解决这个问题。

减小负载均衡器运行状况检查阈值。负载均衡器在向您的应用程序发送流量之前执行运行状况检查。目标组的默认运行状况检查配置可能需要 90 秒或更长时间。在此期间，负载均衡器会检查运行状况，并接收请求。降低运行状况检查间隔和阈值计数可以使您的应用程序更快接受流量并减少其他任务的负载。

请考虑冷启动性能。有些应用程序使用运行时，例如 Java 执行即时（JIT）编译。编译过程至少可以在开始时显示应用程序的性能。一种解决方法是用不会造成冷启动性能损失的语言重新编写工作负载中延迟关键的部分。

使用步进缩放，而不是目标跟踪缩放策略。Amazon ECS 任务有数个 Application Auto Scaling 选项。目标跟踪是最易于使用的模式。有了它，您所需要做的就是为指标设置一个目标值，例如 CPU 平均利用率。然后，自动定标器会自动管理实现该值所需的任务数量。使用分步扩展，您可以更快地对需求变化做出反应，因为您可以定义扩展指标的特定阈值，以及在超过阈值时要添加或删除的任务数量。并

且，更重要的是，您可以通过最大限度地减少突破阈值警报的时间来对需求变化做出快速反应。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的服务 Auto Scaling。

如果您使用 Amazon EC2 实例提供集群容量，则请考虑以下建议：

使用更大的 Amazon EC2 实例和更快的 Amazon EBS 卷。您可以使用更大的 Amazon EC2 实例和更快的 Amazon EBS 卷来提高映像下载和准备速度。在给定的 Amazon EC2 实例系列中，网络和 Amazon EBS 最大吞吐量会随着实例大小的增加而增加（例如，从 `m5.xlarge` 到 `m5.2xlarge`）。此外，您也可以自定义 Amazon EBS 卷以提高其吞吐量和 IOPS。例如，如果您使用的是 `gp2` 卷，则请使用能提供更高基准吞吐量的更大的卷。如果您使用的是 `gp3` 卷，则请在创建卷时指定吞吐量和 IOPS。

请对在 Amazon EC2 实例上运行的任务使用桥式网络模式。在 Amazon EC2 上使用 `bridge` 网络模式的任务比使用 `awsvpc` 网络模式的任务启动得更快。使用 `awsvpc` 网络模式时，Amazon ECS 会在启动任务之前将弹性网络接口（ENI）附加到实例。这样会带来额外的延迟。不过，使用桥式网络需要进行几项权衡。这些任务没有自己的安全组，并且会对负载均衡产生一些影响。有关更多信息，请参阅《弹性负载均衡用户指南》中的负载均衡器目标组。

处理需求冲击

一些应用程序会突然经受巨大的需求冲击。发生这种情况的原因有很多：新闻事件、大促销、媒体事件或其他一些可以使流量在很短的时间内迅速显著增加的病毒式传播事件。如果是意外发生的，则可能会导致需求迅速超过可用资源。

处理需求冲击的最佳方法是预测它们并做出相应的计划。由于自动扩缩可能需要时间，因此建议您在需求冲击开始之前横向扩展应用程序。为了获得最佳结果，建议制定一项业务计划，其中包括使用共享日历的团队之间的紧密协作。计划活动的团队应提前与负责应用程序的团队密切合作。这使得该团队有足够的时间来制定明确的安排计划。他们可以安排容量，以便在活动开始前进行横向扩展，在活动结束后进行横向缩减。有关更多信息，请参阅《Application Auto Scaling 用户指南》中的计划的扩展。

如果您有企业级支持计划，则请同时确保与您的技术客户经理（TAM）合作。您的 TAM 可以验证您的服务配额，并确保在活动开始之前提高所有必要的配额。这样，您将不会意外达到任何服务配额。他们还可以通过预热服务（如负载均衡器等）来帮助您，以确保您的活动顺利进行。

处理计划外的需求冲击是一个更为困难的问题。计划外的冲击，如果幅度足够大，则会迅速导致需求超出容量。它还可以超越自动扩缩的反应能力。应对计划外冲击的最佳方法是过度配置资源。您必须拥有足够的资源来随时处理最大的预期流量需求。

在预期会出现计划外的需求冲击时保持最大容量可能会付出高昂的代价。为了缓解成本影响，请找到预测大规模需求冲击即将来临的领先指标或事件。如果指标或事件可靠地提供了重要的提前通知，则在事件发生或指标超过您设置的特定阈值时，立即开始横向扩展流程。

如果您的应用程序容易出现突然的计划外需求冲击，则请考虑在应用程序中添加一种高性能模式，该模式会牺牲掉非关键功能，但为客户保留关键功能。例如，假设您的应用程序可以切换，从生成昂贵的自定义响应切换到提供静态响应页面。在这种情况下，您可以在不扩缩应用程序的情况下显著提高吞吐量。

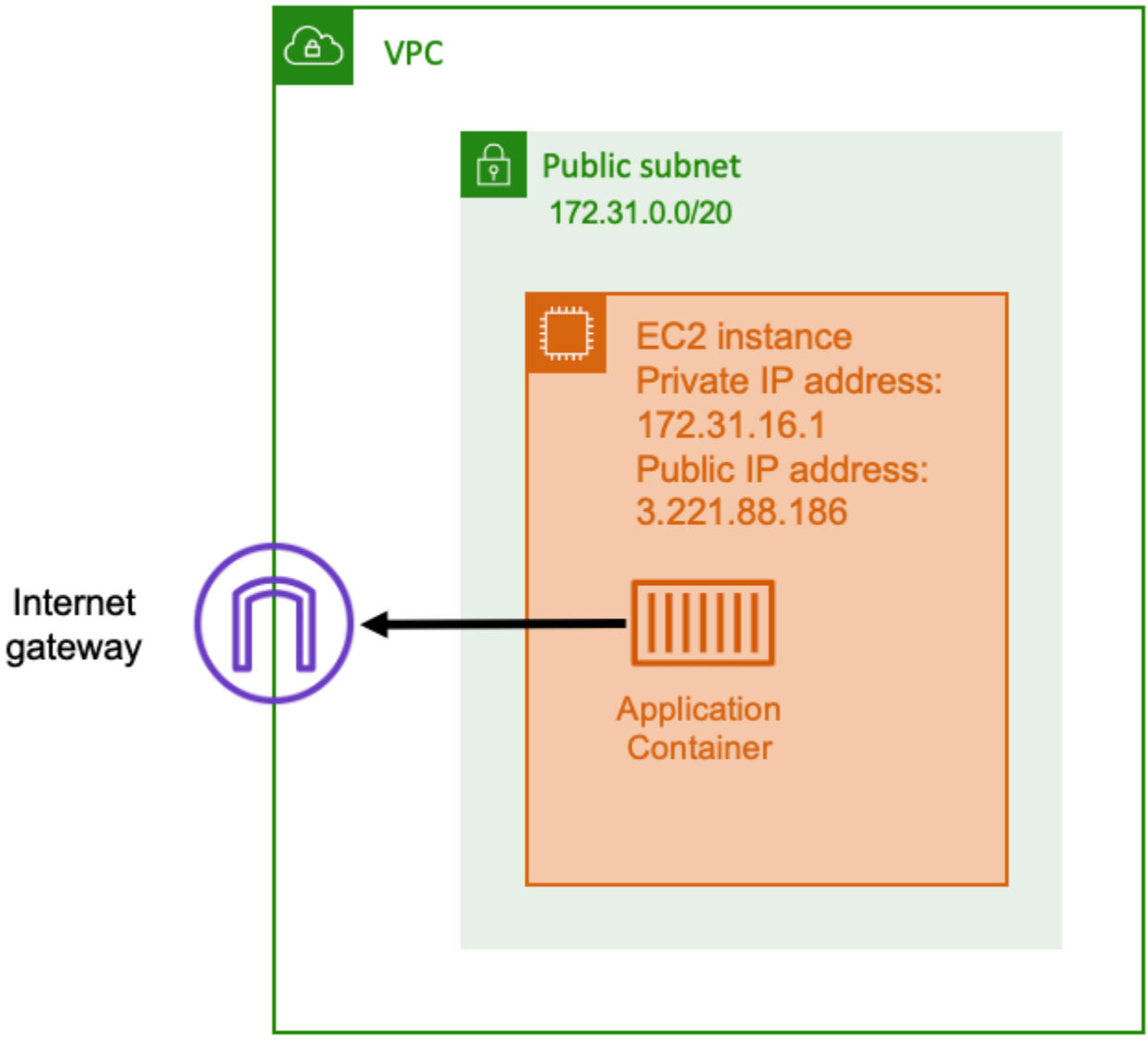
最后，您可以考虑拆分单体服务，以更好地应对需求冲击。如果您的应用程序是一项运行成本高昂且扩展速度较慢的单体服务，则可以提取或重写性能关键部分并将其作为单独的服务运行。然后，这些新服务可以独立于不太关键的组件进行扩展。能够灵活地将性能关键型功能与应用程序的其他部分分开横向扩展，既可以减少增加容量所需的时间，又有助于节省成本。

将 Amazon ECS 应用程序连接到互联网

大多数容器化应用程序至少有一些组件需要出站访问互联网。例如，移动应用的后端需要出站访问权限才能推送通知。

Amazon Virtual Private Cloud 有两种主要方法可促进您的 VPC 与互联网之间的通信。

公有子网和互联网网关



当您使用具有到互联网网关的路由的公有子网时，您的容器化应用程序可以在公有子网上 VPC 内的主机上运行。运行容器的主机分配了公有 IP 地址。此公共 IP 地址可从互联网路由。有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

这种网络架构便于运行您的应用程序的主机与互联网上的其他主机之间的直接通信。通信是双向的。这意味着不仅您可以与互联网上的任何其他主机建立出站连接，而且互联网上的其他主机也可尝试连接到您的主机。因此，您应该密切关注安全组和防火墙规则。这样可以确保互联网上的其他主机无法打开任何您不想打开的连接。

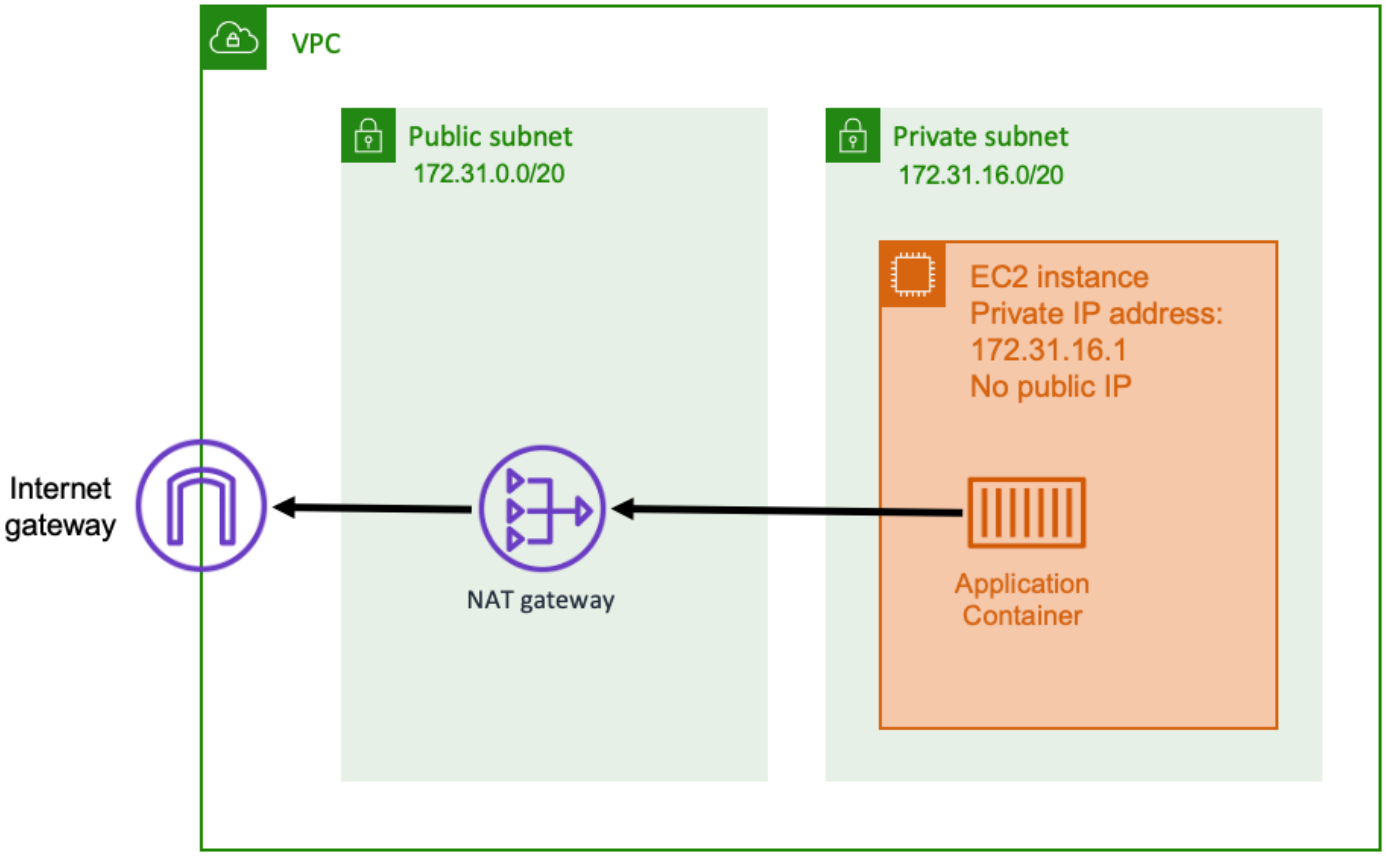
例如，如果您的应用程序在 Amazon EC2 上运行，则请确保未打开用于 SSH 访问的端口 22。否则，您的实例可能会不断收到来自互联网上恶意机器人的 SSH 连接尝试。这些机器人通过公有 IP 地址拖网。其找到一个开放的 SSH 端口后，会尝试暴力破解密码以尝试访问您的实例。因此，许多组织限制了公有子网的使用，并倾向于将大部分（如果并非全部）资源放在私有子网中。

使用公有子网进行联网适用于需要大量带宽或最小延迟的公共应用程序。适用的使用案例包括视频流和游戏服务。

当您在 Amazon EC2 上使用 Amazon ECS 以及在 AWS Fargate 上使用 Amazon ECS 时，都支持这种联网方法。

- Amazon EC2 – 您可以在公有子网上启动 EC2 实例。Amazon ECS 使用这些 EC2 实例作为集群容量，在这些实例上运行的任何容器都可以使用主机的底层公有 IP 地址进行出站联网。这适用于 `host` 和 `bridge` 网络模式。但 `awsvpc` 网络模式不提供具有公有 IP 地址的任务 ENI。因此，他们不能直接使用互联网网关。
- Fargate – 创建 Amazon ECS 服务时，为服务的联网配置指定公有子网，并使用分配公有 IP 地址选项。每个 Fargate 任务都在公有子网中联网，并有自己的公有 IP 地址，用于与互联网直接通信。

私有子网和 NAT 网关



使用私有子网和 NAT 网关时，可以在私有子网中的主机上运行容器化应用程序。因此，该主机的私有 IP 地址可在您的 VPC 内路由，但不能从互联网进行路由。这意味着 VPC 内的其他主机可以使用其私有 IP 地址连接到该主机，但是互联网上的其他主机无法与该主机进行任何入站通信。

借助私有子网，您可以使用网络地址转换（NAT）网关允许私有子网中的主机连接到互联网。互联网上的主机收到的入站连接显示为来自公有子网内 NAT 网关的公有 IP 地址。NAT 网关负责充当互联网和私有 VPC 之间的桥梁。出于安全考虑，通常首选这种配置，因为其意味着您的 VPC 受到保护，不会被互联网上的攻击者直接访问。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT 网关。

这种私有联网方法适用于您想要保护容器免于外部直接访问的场景。适用场景包括支付处理系统或存储用户数据和密码的容器。您在账户中创建和使用 NAT 网关会产生费用。NAT 网关小时使用费率和数据处理费率也适用。出于冗余目的，您应在每个可用区中有一个 NAT 网关。这样，单个可用区的失去可用性不会影响您的出站连接。因此，如果您的工作负载较小，则使用私有子网和 NAT 网关可能更经济实惠。

在 Amazon EC2 上使用 Amazon ECS 以及在 AWS Fargate 上使用 Amazon ECS 时，都支持这种联网方法。

- Amazon EC2 – 您可以在私有子网上启动 EC2 实例。在这些 EC2 主机上运行的容器使用底层主机联网，出站请求通过 NAT 网关传输。
- Fargate – 创建 Amazon ECS 服务时，为服务的联网配置指定私有子网，并且不使用分配公有 IP 地址选项。每个 Fargate 任务都在私有子网中托管。其出站流量通过您与该私有子网相关联的任何 NAT 网关进行路由。

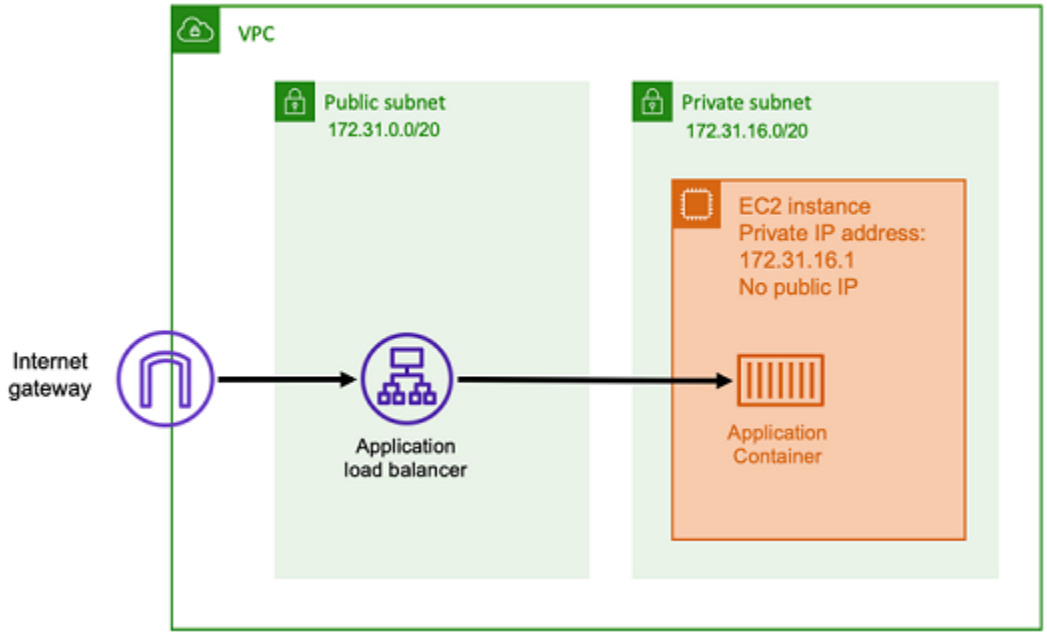
从互联网接收到 Amazon ECS 的入站连接的最佳实践

如果您运行公共服务，则必须接受来自互联网的入站流量。例如，您的公共网站必须接受来自浏览器的入站 HTTP 请求。在这种情况下，互联网上的其他主机也必须启动到您的应用程序主机的入站连接。

解决此问题的一种方法是，在具有公有 IP 地址的公有子网中的主机上启动容器。但是，不建议对大型应用程序使用此方法。对于这些应用程序，更好的方法是在互联网与您的应用程序之间建立一个可扩展的输入层。对于这种方法，您可以使用本节中列出的任何 AWS 服务作为输入。

应用程序负载均衡器

应用程序负载均衡器在应用程序层正常运行。这是开放系统互连（OSI）模型的第 7 层。这使得应用程序负载均衡器适合公共 HTTP 服务。如果您有网站或 HTTP REST API，则应用程序负载均衡器就是适合此工作负载的负载均衡器。有关更多信息，请参阅《应用程序负载均衡器用户指南》中的什么是应用程序负载均衡器？。



使用此架构，您可以在公有子网中创建应用程序负载均衡器，使其具有公有 IP 地址并可以接收来自互联网的入站连接。当应用程序负载均衡器收到入站连接（或者更具体地说，HTTP 请求）时，它将使用其私有 IP 地址打开与应用程序的连接。然后，其通过内部连接转发请求。

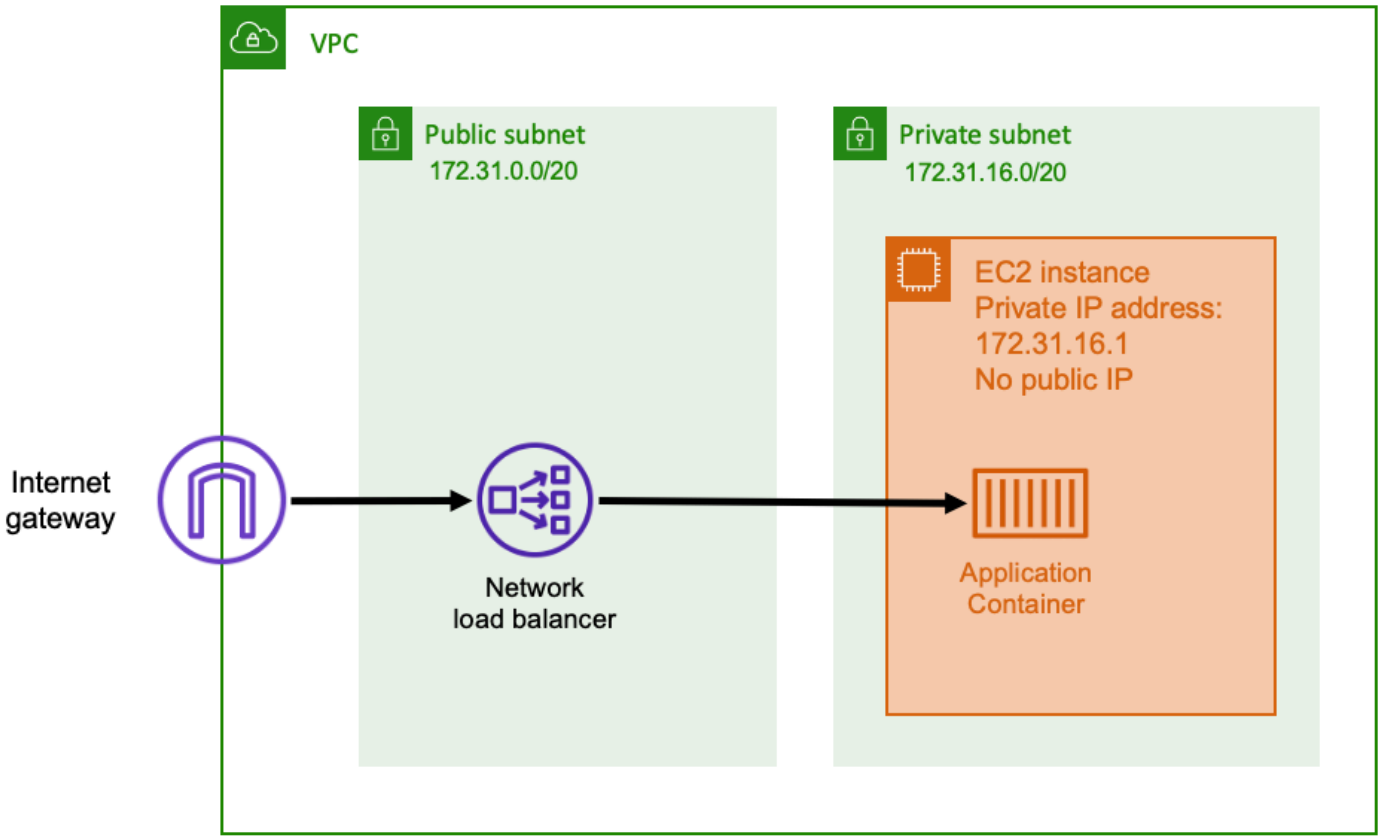
应用程序负载均衡器有以下优势。

- SSL/TLS 终止 – 应用程序负载均衡器可以维持安全的 HTTPS 通信和证书，以便与客户端通信。它可以选择在负载均衡器级别终止 SSL 连接，这样您就不必在自己的应用程序中处理证书了。
- 高级路由 – 应用程序负载均衡器可以有多个 DNS 主机名。它还具有高级路由功能，可以根据请求的主机名或路径等指标将传入的 HTTP 请求发送到不同的目的地。这意味着您可以使用单个应用程序负载均衡器作为许多不同内部服务的输入，甚至是 REST API 不同路径上微服务的输入。
- gRPC 支持和 Websocket – 应用程序负载均衡器可以处理的不仅仅是 HTTP。它还可以对 gRPC 和基于 Websocket 的服务进行负载均衡，并支持 HTTP/2。
- 安全 – 应用程序负载均衡器可帮助保护您的应用程序免受恶意流量的侵害。它包括 HTTP 取消同步缓解等功能，并与 AWS Web 应用程序防火墙（AWS WAF）集成。AWS WAF 可以进一步过滤掉可能包含攻击模式的恶意流量，例如 SQL 注入或跨站脚本攻击。

网络负载均衡器

网络负载均衡在开放系统互连 (OSI) 模型的第四层运行。它适用于非 HTTP 协议或需要端到端加密的场景，但不具有与应用程序负载均衡器相同的 HTTP 特定功能。因此，网络负载均衡器最适合不使

用 HTTP 的应用程序。有关更多信息，请参阅《网络负载均衡器用户指南》中的什么是网络负载均衡器？。



当使用网络负载均衡器作为输入时，其功能与应用程序负载均衡器类似。这是因为它是在公有子网中创建的，并且具有可在互联网上访问的公有 IP 地址。然后，网络负载均衡器会打开与运行容器的主机私有 IP 地址的连接，并将数据包从公共端发送到私有端。

网络负载均衡器功能

由于网络负载均衡器在较低的网络堆栈级别运行，因此它不具备与应用程序负载均衡器相同的功能集。但是，它确实具有以下重要功能。

- 端到端加密 – 由于网络负载均衡器在 OSI 模型的第四层运行，因此它不会读取数据包的内容。这使其适用于需要端到端加密的负载均衡通信。
- TLS 加密 – 除了端到端加密外，网络负载均衡器还可以终止 TLS 连接。这样，您的后端应用程序就不必实现自己的 TLS 了。
- UDP 支持 – 由于网络负载均衡器在 OSI 模型的第四层运行，因此它适用于非 HTTP 工作负载和 TCP 以外的协议。

关闭连接

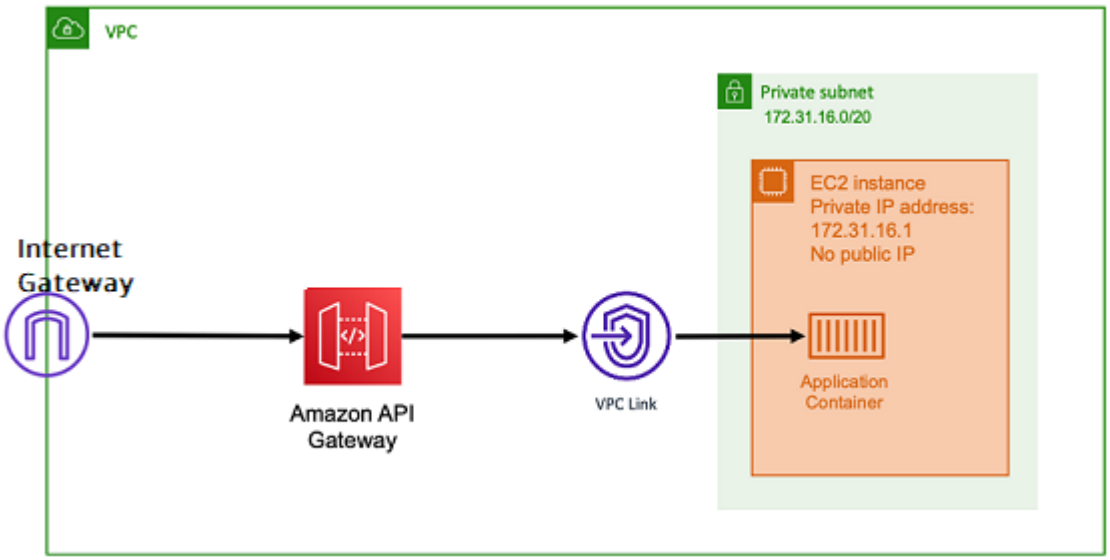
由于网络负载均衡器不遵循 OSI 模型更高层的应用程序协议，因此它无法使用这些协议向客户端发送关闭消息。与应用程序负载均衡器不同，这些连接需要由应用程序关闭；您也可以将网络负载均衡器配置为在任务停止或替换时关闭第四层连接。请参阅网络负载均衡器文档中网络负载均衡器目标组的连接终止设置。

如果客户端不进行处理，让网络负载均衡器关闭第四层连接可能会导致客户端显示不需要的错误消息。有关推荐的客户端配置的更多信息，请在此处参阅生成器库。

关闭连接的方法因应用程序而异，但有一种方法是确保网络负载均衡器目标注销延迟的时间比客户端连接超时长。客户端将首先超时，然后通过网络负载均衡器正常重新连接到下一个任务，而旧任务会慢慢耗尽所有客户端。有关网络负载均衡器目标注销延迟的更多信息，请参阅网络负载均衡器文档。

Amazon API Gateway HTTP API

Amazon API Gateway 适用于请求量突然激增或请求量较低的 HTTP 应用程序。有关更多信息，请参阅《API Gateway 开发人员指南》中的什么是 Amazon API Gateway？。



应用程序负载均衡器和网络负载均衡器的定价模型都包括每小时价格，以使负载均衡器随时可以接受传入的连接。相比之下，API Gateway 会对每个请求单独收费。其效果是，如果没有收到请求，则不收取任何费用。在高流量负载下，应用程序负载均衡器或网络负载均衡器能够以比 API Gateway 更便宜的每次请求价格处理更大的请求量。但是，如果您的请求总数较少或流量较低，则使用 API Gateway 的累积价格应该比按小时收费来维护未得到充分利用的负载均衡器更经济实惠。API Gateway 还可以缓存 API 响应，这可能会降低后端请求速率。

API Gateway 功能使用 VPC 链接，允许 AWS 托管服务使用其私有 IP 地址连接到 VPC 私有子网内的主机。它可以通过查看由 Amazon ECS 服务发现管理的 AWS Cloud Map 服务发现记录来检测这些私有 IP 地址。

API Gateway 支持以下功能。

- API Gateway 操作与负载均衡器类似，但具有 API 管理独有的附加功能
- API Gateway 提供了有关客户端授权、使用套餐和请求/响应修改的其他功能。有关更多信息，请参阅 Amazon API Gateway 功能。
- API Gateway 可以支持边缘、区域和私有 API 网关端点。边缘端点可通过托管的 CloudFront 分配获得。区域端点和私有端点均为区域本地。
- SSL/TLS 终止
- 将不同的 HTTP 路径路由到不同的后端微服务

除了上述功能外，API Gateway 还支持使用自定义 Lambda 授权方，您可以使用这种方法保护您的 API 免于未经授权的使用。有关更多信息，请参阅 Field Notes: Serverless Container-based APIs with Amazon ECS and Amazon API Gateway。

通过账户设置访问 Amazon ECS 功能

您可以进入 Amazon ECS 账户设置以选择使用或选择不使用特定功能。对于每个 AWS 区域，您可以在账户级别或为特定用户或角色选择使用或选择不使用每个账户设置。

如果以下任何一项与您相关，您可能希望选择使用或选择不使用特定功能：

- 用户或角色可以为其个人账户选择使用或选择不使用特定账户设置。
- 用户或角色可以为账户上的所有用户设置默认选择使用或选择不使用设置。
- 根用户或具有管理员权限的用户可以选择使用或选择不使用账户上的任何特定角色或用户。如果根用户的账户设置发生更改，则会为未选择个人账户设置的所有用户和角色设置默认值。

> **Note**
>
> 联合用户采用根用户的账户设置，并且不能单独为他们设置显式账户设置。

可供使用的账户设置如下：您必须单独选择使用还是不使用每个账户设置。

Amazon Resource Names (ARN) 和 ID

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 Resource Name (ARN) 和资源 ID 的新格式。每种资源类型的选择使用状态决定资源使用的 Amazon 资源名称 (ARN) 格式。您必须选择使用新的 ARN 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 Resource Names (ARN) 和 ID。

默认为 `enabled`。

只有在选择进入后启动的资源才能接收新的 ARN 和资源 ID 格式。所有现有资源均不受影响。要使 Amazon ECS 服务和任务转换为新的 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，必须耗尽该容器实例，然后启动并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。

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。您可以使用此配置在每个容器实例上使用 `awsvpc` 网络模式放置更多任务。使用此功能，启用 `awsvpcTrunking` 的 `c5.large` 实例的 ENI 限额增加了 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。主网络接口和中继网络接口不计入 ENI 限额。因此，您可以使用此配置在容器实例上启动 10 个任务，而不是当前的两个任务。有关更多信息，请参阅 增加 Amazon ECS Linux 容器实例网络接口。

默认为 `disabled`。

只有在选择加入后启动的资源才会收到增加的 ENI 限额。所有现有资源均不受影响。要将容器实例转换为增加的 ENI 限额，必须耗尽该容器实例，然后向集群注册一个新的容器实例。

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 使用 Container Insights 监控 Amazon ECS 容器。

当您选择使用 `containerInsights` 账户设置时，预设情况下，所有新集群都启用了容器见解。您可以在创建特定群集时对其禁用此设置。也可以使用更新集群设置 API 更改此设置。

对于包含使用 EC2 启动类型的任务或服务的集群，容器实例必须运行 1.29.0 或更高版本的 Amazon ECS 代理才能使用 Container Insights。有关更多信息，请参阅 Amazon ECS Linux 容器实例管理。

默认为 `disabled`。

双堆栈 VPC IPv6

资源名称：`dualStackIPv6`

Amazon ECS 支持使用 IPv4 地址以外的 IPv6 地址提供任务。

要接收 IPv6 地址的任务，任务必须使用 `awsvpc` 网络模式，必须在配置为双堆栈模式的 VPC 中启动，`dualStackIPv6` 账户设置必须已启用。有关其他要求的更多信息，请参阅 在双堆栈模式下使用 VPC（EC2 启动类型）和 在双堆栈模式下使用 VPC（Fargate 启动类型）。

> **Important**
>
> `dualStackIPv6` 账户设置只能使用 Amazon ECS API 或 AWS CLI 进行更改。有关更多信息，请参阅 修改 Amazon ECS 账户设置。

如果您在 2020 年 10 月 1 日至 2020 年 11 月 2 日期间在启用 IPv6 的子网中使用 `awsvpc` 网络模式运行任务，则运行该任务的区域中的默认 `dualStackIPv6` 账户设置为 `disabled`。如果不满足该条件，则区域中的默认 `dualStackIPv6` 设置为 `enabled`。

默认为 `disabled`。

Fargate FIPS-140 合规性

资源名称：`fargateFIPSMode`

Fargate 支持美国联邦信息处理标准（FIPS-140），其中规定了对保护敏感信息的加密模块的安全要求。它是美国和加拿大政府的现行标准，适用于需要符合《联邦信息安全管理法》（FISMA）或联邦风险和授权管理计划（FedRAMP）的系统。

默认为 `disabled`。

您必须开启 FIPS-140 合规性。有关更多信息，请参阅 the section called “AWS Fargate FIPS-140 合规性”。

> **Important**
>
> `fargateFIPSMode` 账户设置只能使用 Amazon ECS API 或 AWS CLI 进行更改。有关更多信息，请参阅 修改 Amazon ECS 账户设置。

标记资源授权

资源名称：`tagResourceAuthorization`

某些 Amazon ECS API 操作允许您在创建资源时指定标签。

Amazon ECS 正在为资源创建引入标记授权。用户必须具有创建资源的操作的权限，例如 `ecsCreateCluster`。如果在资源创建操作中指定了标签，则 AWS 会对 `ecs:TagResource` 操作执行额外的授权，以验证用户或角色是否具备创建标签的权限。因此，您必须授予使用 `ecs:TagResource` 操作的显式权限。有关更多信息，请参阅 the section called “在创建过程中，为资源添加标签”。

Fargate 任务停用等待期

资源名称：`fargateTaskRetirementWaitPeriod`

AWS 负责修补和维护 AWS Fargate 的底层基础设施。当 AWS 确定 Fargate 上托管的 Amazon ECS 任务需要进行安全或基础设施更新时，需要停止这些任务并启动新任务来替换它们。您可以对停用任务进行修补之前的等待时间进行配置。您可以选择立即停用该任务、等待 7 个日历日或等待 14 个日历日。

此设置于账户级别进行。

运行时监控激活

资源名称：`guardDutyActivate`

`guardDutyActivate` 参数在 Amazon ECS 中是只读的，指示安全管理员在您的 Amazon ECS 账户中开启还是关闭运行时监控。GuardDuty 代表您控制此账户设置。有关更多信息，请参阅使用运行时监控保护 Amazon ECS 工作负载。

主题

- Amazon Resource Names (ARN) 和 ID
- ARN 和资源 ID 格式时间表
- AWS Fargate 美国联邦信息处理标准（FIPS-140）合规性
- 标记授权
- 标记授权时间表
- AWS Fargate 任务停用等待时间
- 运行时监控（Amazon GuardDuty 集成）
- 使用控制台查看 Amazon ECS 账户设置
- 修改 Amazon ECS 账户设置
- 恢复到默认 Amazon ECS 账户设置
- 使用 AWS CLI 管理 Amazon ECS 账户设置

Amazon Resource Names (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 Resource Name (ARN) 和资源标识符 (ID)。如果通过命令行工具或 Amazon ECS API 使用 Amazon EC2，则某些命令需要资源 ARN 或 ID。例如，如果使用 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。

您可以按区域选择使用和选择不使用新的 Amazon 资源名称 (ARN) 和资源 ID 格式。目前，预设情况下，所有创建的新账户都选择使用。

您可以随时选择使用或选择不使用新的 Amazon Resource Name (ARN) 和资源 ID 格式。在选择加入后，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，选择使用或选择不使用新格式不会影响现有的资源 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详细信息，请参阅 Amazon Elastic Container Service 常见问题。

Amazon Resource Name (ARN) 格式

一些资源具有用户友好的名称，例如名为 `production` 的服务。在其他情况下，您必须使用 Amazon Resource Name (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 格式的信息，请参阅 修改 Amazon ECS 账户设置。

下表显示了每个资源类型的当前格式和新格式。

资源类型	ARN
容器实例	当前：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	当前：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	当前：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。当前资源 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 格式的信息，请参阅 修改 Amazon ECS 账户设置。

ARN 和资源 ID 格式时间表

Amazon ECS 资源的新 Amazon 资源名称（ARN）和资源 ID 格式的选择使用和选择不使用周期的时间表于 2021 年 4 月 1 日结束。默认情况下，所有账户都选择使用新格式。所有新创建的资源都会采用新格式，您无法再选择不使用。

AWS Fargate 美国联邦信息处理标准（FIPS-140）合规性

您必须在 Fargate 上开启美国联邦信息处理标准（FIPS-140）合规性。有关更多信息，请参阅 the section called “AWS Fargate FIPS-140 合规性”。

在 `fargateFIPSMode` 选项设置为 `enabled` 的情况下运行 `put-account-setting-default`。有关更多信息，请参阅 Amazon Elastic Container Service API 参考中的 put-account-setting-default。

- 您可以使用以下命令开启 FIPS-140 合规性。

```
aws ecs put-account-setting-default --name fargateFIPSMode --value enabled
```

示例输出

```
{
    "setting": {
        "name": "fargateFIPSMode",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
         "type": user
    }
}
```

您可以运行 `list-account-settings` 以查看当前的 FIPS-140 合规性状态。使用 `effective-settings` 选项以查看账户级别设置。

```
aws ecs list-account-settings --effective-settings
```

标记授权

Amazon ECS 正在为资源创建引入标记授权。用户必须具有创建资源的操作的标记权限，例如 `ecsCreateCluster`。当您创建资源并为该资源指定标签时，AWS 会执行额外授权以验证是否有创

建标签的权限。因此，您必须授予使用 ecs:TagResource 操作的显式权限。有关更多信息，请参阅 the section called “在创建过程中，为资源添加标签”。

要选择加入标记授权，请在 tagResourceAuthorization 选项设置为 enable 的情况下运行 put-account-setting-default。有关更多信息，请参阅 Amazon Elastic Container Service API 参考中的 put-account-setting-default。您可以运行 list-account-settings 以查看当前的标记授权状态。

- 您可以使用以下命令启用标记授权。

```
aws ecs put-account-setting-default --name tagResourceAuthorization --value on --
region region
```

示例输出

```
{
    "setting": {
        "name": "tagResourceAuthorization",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": user
    }
}
```

启用标记授权后，您必须配置相应的权限，以允许用户在资源创建时标记资源。有关更多信息，请参阅 the section called “在创建过程中，为资源添加标签”。

您可以运行 list-account-settings 以查看当前的标记授权状态。使用 effective-settings 选项以查看账户级别设置。

```
aws ecs list-account-settings --effective-settings
```

标记授权时间表

您可以通过运行 list-account-settings 以查看 tagResourceAuthorization 值来确认标记授权是否处于活动状态。当值为 on 时，意味着标记授权在使用中。有关更多信息，请参阅 Amazon Elastic Container Service API 参考中的 list-account-settings。

以下是与标记授权相关的重要日期。

- 2023 年 4 月 18 日 – 引入标记授权。所有新账户和现有账户都必须选择使用该功能。您可以选择开始使用标记授权。选择加入后，您必须授予相应的权限。
- 2024 年 2 月 9 日 到 2024 年 3 月 6 日 – 默认情况下，所有新账户和未受影响的现有账户均已启用标记授权。您可以启用或禁用 `tagResourceAuthorization` 账户设置来验证您的 IAM 策略。

  AWS 已通知受影响的账户。

  要禁用该功能，请在 `tagResourceAuthorization` 选项设置为 `off` 的情况下运行 `put-account-setting-default`。
- 2024 年 3 月 7 日 – 如果您启用了标记授权，则无法再禁用该账户设置。

  建议您在此日期之前完成 IAM 策略测试。
- 2024 年 3 月 29 日 – 所有账户均使用标记授权。Amazon ECS 控制台或 AWS CLI 中将不再提供账户级别设置。

AWS Fargate 任务停用等待时间

当您在标记为停用的平台版本修订版上运行 Fargate 任务时，AWS 会发出通知。有关更多信息，请参阅 Amazon ECS 上的 AWS Fargate 任务维护常见问题。

您可以对 Fargate 开始停用任务的时间进行配置。对于需要立即应用更新的工作负载，请选择即时设置（`0`）。当您需要更多控制时，例如，当任务只能在特定时间段内停止时，请配置 7 天（`7`）或 14 天（`14`）选项。

建议您选择较短的等待时间，以便更快获得较新的平台版本修订版。

通过以根用户或管理员用户身份运行 `put-account-setting-default` 或 `put-account-setting` 配置等待期。将 `fargateTaskRetirementWaitPeriod` 选项用于设置为以下值之一的 `name` 和 `value` 选项：

- `0` - AWS 发送通知，并立即开始停用受影响的任务。
- `7` - AWS 发送通知，等待 7 个日历日后才开始停用受影响的任务。
- `14` - AWS 发送通知，等待 14 个日历日后才开始停用受影响的任务。

默认值为 7 天。

有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 put-account-setting-default 和 put-account-setting。

您可运行以下命令将等待期设置为 14 天。

```
aws ecs put-account-setting-default --name fargateTaskRetirementWaitPeriod --value 14
```

示例输出

```
{
    "setting": {
        "name": "fargateTaskRetirementWaitPeriod",
        "value": "14",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type: user"
    }
}
```

您可以运行 `list-account-settings` 以查看当前 Fargate 任务停用等待时间。使用 `effective-settings` 选项。

```
aws ecs list-account-settings --effective-settings
```

运行时监控（Amazon GuardDuty 集成）

运行时监控是一项智能威胁检测服务，它通过持续监控 AWS 日志和联网活动来识别恶意或未经授权的行为，从而保护在 Fargate 和 EC2 容器实例上运行的工作负载。

`guardDutyActivate` 参数在 Amazon ECS 中是只读的，指示安全管理员在您的 Amazon ECS 账户中开启还是关闭运行时监控。GuardDuty 代表您控制此账户设置。有关更多信息，请参阅使用运行时监控保护 Amazon ECS 工作负载。

您可以运行 `list-account-settings` 以查看当前 GuardDuty 集成设置。

```
aws ecs list-account-settings
```

示例输出

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "on",
```

```
        "principalArn": "arn:aws:iam::123456789012:doej",
        "type": aws-managed"
    }
}
```

使用控制台查看 Amazon ECS 账户设置

您可以使用 AWS Management Console 查看您的账户设置。

> **Important**
>
> 只能使用 AWS CLI 查看或更改 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 账户设置。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在顶部的导航栏中，选择要查看其账户设置的区域。
3. 在导航页面中，选择 Account Settings（账户设置）。

修改 Amazon ECS 账户设置

您可以使用 AWS Management Console 修改您的账户设置。

`guardDutyActivate` 参数在 Amazon ECS 中是只读的，指示安全管理员在您的 Amazon ECS 账户中开启还是关闭运行时监控。GuardDuty 代表您控制此账户设置。有关更多信息，请参阅使用运行时监控保护 Amazon ECS 工作负载。

> **Important**
>
> 只能使用 AWS CLI 查看或更改 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 账户设置。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在顶部的导航栏中，选择要查看其账户设置的区域。
3. 在导航页面中，选择 Account Settings（账户设置）。
4. 选择更新。

5. 要增加或减少可为每个 EC2 实例在 awsvpc 网络模式下运行的任务数量，请在AWSVPC 中继下选择AWSVPC 中继。
6. 要默认使用或停止使用集群的 CloudWatch Container Insights，请在 CloudWatch Container Insights 下选择或清除 CloudWatch Container Insights。
7. 要启用或禁用标记授权，请在资源标记授权下，选择或清除资源标记授权。
8. 选择 Save changes（保存更改）。
9. 在确认屏幕上，选择 Confirm (确认) 以保存选择。

恢复到默认 Amazon ECS 账户设置

您可以使用 AWS Management Console 将 Amazon ECS 账户设置恢复到默认设置。

仅当您的账户设置不再是默认设置时，Revert to account default（恢复为账户默认值）选项才可用。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在顶部的导航栏中，选择要查看其账户设置的区域。
3. 在导航页面中，选择 Account Settings（账户设置）。
4. 选择更新。
5. 选择 Revert to account default（恢复为账户默认值）。
6. 在确认屏幕上，选择 Confirm (确认) 以保存选择。

使用 AWS CLI 管理 Amazon ECS 账户设置

您可以使用 Amazon ECS API、AWS CLI 或 SDK 管理账户设置。只能使用这些工具查看或更改 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 账户设置。

有关任务定义的可用 API 操作的信息，请参阅《Amazon Elastic Container Service API 参考》中的账户设置操作。

使用下列命令之一修改账户中所有 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
```

您还可以使用此命令修改其他账户设置。为此，请将 name 参数替换为相应的账户设置。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-east-1 -Force
```

修改用户账户的账户设置 (AWS CLI)

使用下列命令之一修改您的 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，除非某个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
```

您还可以使用此命令修改其他账户设置。为此，请将 name 参数替换为相应的账户设置。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

修改特定用户或角色的账户设置 (AWS CLI)

使用下列命令之一，在请求中指定用户、角色或根用户的 ARN，以修改特定用户或角色的账户设置。

- 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
```

您还可以使用此命令修改其他账户设置。为此，请将 name 参数替换为相应的账户设置。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

适用于 Amazon ECS 的 IAM 角色

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。在 Amazon ECS 中，您可以创建角色来授予对 Amazon ECS 资源（例如容器或服务）的权限。

Amazon ECS 需要的角色取决于任务定义启动类型和您使用的功能。使用下表确定 Amazon ECS 需要哪些 IAM 角色。

角色	定义	何时需要	更多信息
任务执行角色	此角色允许 Amazon ECS 代表您使用其他 AWS 服务。	您的任务托管在 AWS Fargate 或外部实例上且： • 从 Amazon ECR 私有存储库中提取容器映像。 • 在与运行任务的账户不同的账户中，从 Amazon ECR 私有存储库中提取容器映像。 • 使用 `awslogs` 日志驱动程序将容器日志发送到 CloudWatch Logs。 您的任务托管在 AWS Fargate 或 Amazon EC2 实例上且： • 使用私有注册表身份验证。 • 使用运行时监控。 • 任务定义使用 Secrets Manager 密钥或 AWS Systems	Amazon ECS 任务执行 IAM 角色

角色	定义	何时需要	更多信息
		Manager Parameter Store 参数引用敏感数据。	
任务角色	此角色允许您的应用程序代码（在容器上）使用其他 AWS 服务。	您的应用程序访问其他 AWS 服务，例如 Amazon S3。	Amazon ECS 任务 IAM 角色
容器实例角色	此角色允许您的 EC2 实例或外部实例向集群注册。	您的任务托管在 Amazon EC2 实例或外部实例上。	Amazon ECS 容器实例 IAM 角色
Amazon ECS Anywhere 角色	此角色允许您的外部实例访问 AWS API。	您的任务托管在外部实例上。	Amazon ECS Anywhere IAM 角色
Amazon ECS CodeDeploy 角色	此角色允许 CodeDeploy 对您的服务进行更新。	您可使用 CodeDeploy 蓝/绿部署类型来部署服务。	Amazon ECS CodeDeploy IAM 角色
Amazon ECS EventBridge 角色	此角色允许 EventBridge 对您的服务进行更新。	您可使用 EventBridge 规则和目标来计划任务。	Amazon ECS EventBridge IAM 角色

角色	定义	何时需要	更多信息
Amazon ECS 基础设施角色	此角色允许 Amazon ECS 管理集群中的基础设施资源。	• 您想将 Amazon EBS 卷附加到您的 Fargate 或 EC2 启动类型 Amazon ECS 任务。基础设施角色允许 Amazon ECS 为您的任务管理 Amazon EBS 卷。 • 您想使用传输层安全性协议（TLS）加密您的 Amazon ECS Service Connect 服务之间的流量。	Amazon ECS 基础设施 IAM 角色

Amazon ECS 任务定义

任务定义是应用程序的蓝图。它是描述构成应用程序的参数和一个或多个容器的 JSON 格式的文本文件。

您可在任务定义中指定的一些参数：

- 要使用的启动类型，这决定了托管您的任务的基础设施
- 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
- 内存和 CPU 要求
- 运行任务所在的容器的操作系统
- 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，任务是否继续运行
- 容器在启动时运行的命令
- 在任务中用于容器的任何数据卷
- 您的任务使用的 IAM 角色

要获得任务定义参数的完整列表，请参阅 Amazon ECS 任务定义参数。

创建任务定义后，您可以将任务定义作为任务或服务运行。

- 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。
- Amazon ECS 服务在 Amazon ECS 集群中同时运行和维护您所需数量的任务。它的工作原理是，如果您的任何任务出于任何原因失败或停止，Amazon ECS 服务调度器将根据您的任务定义启动另一个实例。这样做是为了替换它，从而保持服务中所需的任务数量。

主题

- Amazon ECS 任务定义状态
- 设计适用于 Amazon ECS 的应用程序架构
- 使用控制台创建 Amazon ECS 任务定义
- 使用控制台更新 Amazon ECS 任务定义

- 使用控制台注销 Amazon ECS 任务定义修订
- 使用控制台删除 Amazon ECS 任务定义修订
- Amazon ECS 任务定义应用场景
- Amazon ECS 任务定义参数
- Amazon ECS 任务定义模板
- Amazon ECS 任务定义示例

Amazon ECS 任务定义状态

当您创建、取消注册或删除任务定义时，其状态会发生变化。您可以在控制台中或使用 `DescribeTaskDefinition` 查看任务定义状态。

以下是任务定义可能的状态：

ACTIVE

在向 Amazon ECS 注册之后，任务定义处于 `ACTIVE` 状态。您可以使用处于 `ACTIVE` 状态的任务定义来运行任务或创建服务。

INACTIVE (非活跃)

当您取消注册任务定义时，任务定义会从 `ACTIVE` 状态转换为 `INACTIVE` 状态。您可以通过调用 `DescribeTaskDefinition` 来检索 `INACTIVE` 任务定义。您不能使用处于 `INACTIVE` 状态的任务定义来运行新任务或创建新服务。这对现有服务或任务没有影响。

DELETE_IN_PROGRESS

在您提交要删除的任务定义后，任务定义会从 `INACTIVE` 状态转换为 `DELETE_IN_PROGRESS` 状态。任务定义处于 `DELETE_IN_PROGRESS` 状态后，Amazon ECS 会定期验证目标任务定义是否未被任何活动任务或部署引用，然后永久删除该任务定义。您不能使用处于 `DELETE_IN_PROGRESS` 状态的任务定义来运行新任务或创建新服务。您可以随时提交要删除的任务定义，而不会对现有任务和服务造成影响。

您可以在控制台中查看处于 `DELETE_IN_PROGRESS` 状态的任务定义，也可以通过调用 `DescribeTaskDefinition` 来检索任务定义。

删除所有 `INACTIVE` 任务定义修订版时，任务定义名称不会显示在控制台中，也不会在 API 中返回。如果任务定义修订版处于 `DELETE_IN_PROGRESS` 状态，任务定义名称会显示在控制台中，并且在 API 中返回。任务定义名称由 Amazon ECS 保留，并且在下次使用该名称创建任务定义时，修订版本会增加。

如果您使用 AWS Config 管理任务定义，AWS Config 会向您收取所有任务定义注册的费用。您只需为取消注册最新的 ACTIVE 任务定义付费。删除任务定义不收取任何费用。有关定价的更多信息，请参阅 AWS Config 定价。

可以阻止删除的 Amazon ECS 资源

当有任何依赖于任务定义修订的 Amazon ECS 资源时，将无法完成任务定义删除请求。以下资源可能会阻止任务定义被删除：

- Amazon ECS 任务 - 需要任务定义才能使任务保持正常运行。
- Amazon ECS 部署和任务集 - 在 Amazon ECS 部署或任务集启动扩展事件时，需要任务定义。

如果您的任务定义仍处于 DELETE_IN_PROGRESS 状态，则可以使用控制台或 AWS CLI 来识别，然后停止阻止删除任务定义的资源。

移除被阻止的资源后删除任务定义

在您移除阻止删除任务定义的资源后，将适用以下规则：

- Amazon ECS 任务 - 任务停止后，删除任务定义最多可能需要 1 小时才能完成。
- Amazon ECS 部署和任务集 - 删除部署或任务集后，任务定义删除最多可能需要 24 小时才能完成。

设计适用于 Amazon ECS 的应用程序架构

您可以通过为应用程序创建任务定义来构建应用程序。任务定义包含用于定义有关应用程序的信息的参数，包括：

- 要使用的启动类型，这决定了托管您的任务的基础设施。

  使用 EC2 启动类型时，您还可以选择实例类型。对于某些实例类型，例如 GPU，您需要设置其他参数。有关更多信息，请参阅 Amazon ECS 任务定义应用场景。
- 容器映像，包含您的应用程序代码和运行应用程序代码所需的所有依赖项。
- 在您的任务中用于容器的联网模式

  联网模式决定了任务通过网络进行通信的方式。

对于在 EC2 实例上运行的任务，有多个选项，但建议您使用 awsvpc 网络模式。awsvpc 网络模式可简化容器联网，因为您可以更好地控制您的应用程序相互之间以及与 VPC 内其他服务之间的通信方式。

对于在 Fargate 上运行的任务，您只能使用 awsvpc 网络模式。

- 要用于您的任务的日志记录配置。
- 在任务中用于容器的任何数据卷。

要获得任务定义参数的完整列表，请参阅 Amazon ECS 任务定义参数。

创建任务定义时，请遵循以下准则：

- 将每个任务定义系列仅用于一个业务目的。

  如果您在同一个任务定义中将多种类型的应用程序容器分组在一起，则无法独立扩展这些容器。例如，网站和 API 不太可能要求以相同的速度进行横向扩展。随着流量的增加，所需的 Web 容器数量将与 API 容器数量不同。如果这两个容器部署在同一个任务定义中，则每个任务会运行相同数量的 Web 容器和 API 容器。
- 将每个应用程序版本与任务定义系列中的任务定义修订版进行匹配。

  在任务定义系列中，将每个任务定义修订版视为特定容器映像设置的时间点快照。这类似于容器是一个快照，其中包含运行特定版本应用程序代码所需的所有内容。

  确保应用程序代码版本、容器映像标签和任务定义修订版之间存在一对一的映射。典型的发布过程涉及一个 git commit，它会变成一个标有 git commit SHA 的容器映像。然后，该容器映像标签将获得自己的 Amazon ECS 任务定义修订版。最后，对 Amazon ECS 服务进行了更新，告诉它部署新的任务定义修订版。
- 为每个任务定义系列使用不同的 IAM 角色。

  使用自己的 IAM 角色定义每个任务定义。该建议应与我们的建议同时提出，以为每个业务组件提供自己的任务定义系列。通过实施这两种最佳实践，您可以限制每项服务对您 AWS 账户中资源的访问权限。例如，您可以授予身份验证服务访问权限，以连接到您的密码数据库。同时，您还可以确保只有您的订单服务才能访问信用卡付款信息。

Amazon ECS 容器映像的最佳实践

容器映像是一组关于如何构建容器的说明。容器映像包含您的应用程序代码和运行应用程序代码所需的所有依赖项。应用程序依赖项包括您的应用程序代码所依赖的源代码包、解释型语言的语言运行时系统以及动态链接代码所依赖的二进制包。

在设计和构建容器映像时，请遵循以下指南：

- 通过将所有应用程序依赖项作为静态文件存储在容器映像中，使您的容器映像完整。

  如果更改容器映像中的某些内容，则请使用更改来构建一个新的容器映像。
- 在容器内运行单个应用程序进程。

  容器的生命周期与应用程序进程运行的时间一样长。Amazon ECS 将替换崩溃的进程并确定在何处启动替换进程。完整的映像使整体部署更具弹性。
- 让您的应用程序处理 SIGTERM。

  Amazon ECS 停止任务时，它首先向任务发送 SIGTERM 信号通知应用程序需要完成并关闭。然后，Amazon ECS 会发送一条 SIGKILL 消息。当应用程序忽略 SIGTERM 时，Amazon ECS 服务必须等待发送终止该进程的 SIGKILL 信号。

  您需要确定应用程序需要多长时间才能完成其工作，并确保您的应用程序能够处理 SIGTERM 信号。应用程序的信号处理需要阻止应用程序执行新工作并完成正在进行的工作，或者当工作需要太长时间才能完成时，将未完成的工作保存到任务之外的存储中。
- 配置容器化应用程序以向 stdout 和 stderr 写入日志。

  将日志处理与应用程序代码分离时，可以灵活地在基础设施级别调整日志处理。此操作的一个示例是更改您的日志记录系统。您可以调整设置，而不是修改服务以及构建和部署新的容器映像。
- 使用标签对容器映像进行版本控制。

  容器映像存储在容器注册表中。注册表中的每个映像都由标签标识。有一个名为 latest 的标签。此标签的作用是指向最新版本的应用程序容器映像的指针，类似于 git 存储库的 HEAD 中。建议您仅将此 latest 标签用于测试。最佳做法是，使用每个版本的唯一标签来标记容器映像。建议您使用 git SHA 标记您的映像，以用于构建映像的 git commit。

  您无需为每次提交构建容器映像。但是，建议您在每次发布提交到生产环境的特定代码时都构建一个新的容器映像。还建议您使用与映像内部代码的 git commit 相对应的标签来标记映像。如果您使用 git commit 标记了映像，则可以更快地找到映像正在运行的代码版本。

还建议您在 Amazon Elastic Container Registry 中开启不可变的映像标签。使用此设置，您无法更改标签所指向的容器映像。相反，Amazon ECR 强制要求必须将新映像上传到新标签。有关更多信息，请参阅《Amazon ECR 用户指南》中的映像标签可变性。

构建要在 AWS Fargate 上运行的应用程序时，您必须决定是把多个容器部署到同一个任务定义中，还是分别在多个任务定义中部署容器。当需要满足以下条件时，我们建议您在相同任务定义中部署多个容器：

- 您的容器共享公共生命周期（即，它们应该一起启动和终止）。
- 您的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）。
- 您的容器共享资源。
- 您的容器共享数据卷。

当不需要满足这些条件时，我们建议您在多个任务定义中单独部署容器。这样一来，您可以分别扩展、预调配和取消预调配容器。

Amazon ECS 任务大小的最佳实践

在 Amazon ECS 上部署容器时，要做的最重要的选择之一就是容器和任务大小。容器和任务大小对于扩展和容量计划都至关重要。在 Amazon ECS 中，有两个资源指标用于测量容量：CPU 和内存。CPU 以整个 vCPU 的 1/1024 为单位测量（其中 1024 个单位等于一整个 vCPU）。内存以兆字节为单位测量。在任务定义中，您可以声明资源预留和限制。

当您声明预留时，您就是在声明任务所需的最低资源量。您的任务至少会收到所请求的资源量。您的应用程序可能能够使用比您声明的预留更多的 CPU 或内存。但是，这受制于您所声明的任何限制。使用超过预留量被称为突增。在 Amazon ECS 中，保证预留。例如，如果您使用 Amazon EC2 实例来提供容量，则 Amazon ECS 不会将任务放置在无法完成预留的实例上。

限制是指您的容器或任务可以使用的最大 CPU 单位量或内存量。任何使用超过此限制的 CPU 数量的尝试都会导致限制。任何使用更多内存的尝试都会导致您的容器停止。

选择这些值可能会具有挑战性。这是因为最适合您的应用程序的值在很大程度上取决于应用程序的资源需求。对应用程序进行负载测试是成功计划资源需求和更好地了解应用程序需求的关键。

无状态应用程序

对于水平扩展的无状态应用程序，例如负载均衡器后面的应用程序，建议您首先确定应用程序在提供请求时消耗的内存量。为执行此操作，您可以使用传统工具（例如 `ps` 或 `top`）或监控解决方案（例如 CloudWatch Container Insights）。

在确定 CPU 预留时，请考虑如何扩展应用程序以满足您的业务需求。您可以使用较小的 CPU 预留（例如 256 个 CPU 单元（或 1/4 vCPU），以细粒度的方式进行横向扩展，从而最大限度地降低成本。但是，它们的扩展速度可能不够快，无法满足需求的显著激增。您可以使用更大的 CPU 预留来更快地横向缩减和扩展，从而更快地满足需求激增。但是，预留的 CPU 越大，成本就越高。

其他客户端应用程序

对于无法横向扩展的应用程序（例如单例工作线程或数据库服务器），可用容量和成本是您最重要的考虑因素。您应该根据负载测试表明需要提供的流量来选择内存和 CPU 的数量，以满足您的服务级别目标。Amazon ECS 可确保将应用程序放置在具有足够容量的主机上。

Amazon ECS 的网络安全最佳实践

网络安全是一个包含多个子主题的广泛主题。其中包括传输中加密、网络分段和隔离、防火墙、流量路由和可观测性。

传输中加密

加密网络流量可防止未经授权的用户在通过网络传输数据时拦截和读取数据。使用 Amazon ECS，可以通过以下任何方式实施网络加密。

- 使用服务网格（TLS）：

  使用 AWS App Mesh，您可以在使用网格端点部署的 Envoy 代理之间配置 TLS 连接。虚拟节点和虚拟网关是两个例子。TLS 证书可以来自 AWS Certificate Manager（ACM）。或者，它可以来自您自己的私有证书颁发机构。

  - 启用传输层安全性协议（TLS）
  - 使用 ACM 证书或客户提供的证书启用 AWS App Mesh 中的服务间的流量加密
  - TLS ACM 演练
  - TLS 文件 演练
  - Envoy
- 使用 Nitro 实例：

默认情况下，以下 Nitro 实例类型之间的流量会自动加密：
C5n、G4、I3en、M5dn、M5n、P3dn、R5dn 和 R5n。当流量通过中转网关、负载均衡器或类似中介进行路由时，不会对其进行加密。

- 传输中加密
- 2019 年以来的新公告
- 此演讲来自 re:Inforce 2019

- 将服务器名称指示（SNI）协议与应用程序负载均衡器一起使用：

  应用程序负载均衡器（ALB）和网络负载均衡器（NLB）支持服务器名称指示（SNI）。通过使用 SNI，您可以将多个安全应用程序放在一个侦听器后面。为此，每个侦听器都有自己的 TLS 证书。建议您使用 AWS Certificate Manager（ACM）为负载均衡器预调配证书，然后将其添加到侦听器的证书列表中。AWS 负载均衡器将智能证书选择算法与 SNI 结合使用。如果客户端提供的主机名与证书列表中的一个证书匹配，则负载均衡器将选择此证书。如果客户端提供的主机名与证书列表中的多个证书匹配，则负载均衡器将选择客户端可支持的证书。示例包括自签名证书或通过 ACM 生成的证书。

  - 使用应用程序负载均衡器的 SNI
  - 使用网络负载均衡器的 SNI

- 使用 TLS 证书的端到端加密：

  这涉及使用任务部署 TLS 证书。这可以是自签名证书，也可以是来自可信证书颁发机构的证书。您可以通过引用证书的密钥来获取证书。否则，您可以选择运行一个容器，该容器向 ACM 发出证书签名请求（CSR），然后将生成的密钥挂载到共享卷中。

  - 使用搭载 Amazon ECS 的网络负载均衡器维护整个容器的传输层安全（第 1 部分）
  - 维护整个容器的传输层安全性协议（TLS）第 2 部分：使用 AWS Private Certificate Authority

任务联网

以下建议考虑了 Amazon ECS 的工作原理。Amazon ECS 不使用叠加网络。相反，任务配置为在不同的网络模式下运行。例如，配置为使用 `bridge` 模式的任务会从每台主机上运行的 Docker 网络获取不可路由的 IP 地址。配置为使用 `awsvpc` 网络模式的任务会从主机的子网获取 IP 地址。配置了 `host` 联网功能的任务使用主机的网络接口。`awsvpc` 是首选的网络模式。这是因为它是您可以用来为任务分配安全组的唯一模式。它也是唯一可用于在 Amazon ECS 上的 AWS Fargate 任务的模式。

任务的安全组

建议您配置任务以使用 awsvpc 网络模式。将任务配置为使用此模式后，Amazon ECS 代理会自动预调配弹性网络接口（ENI）并将其附加到该任务。预调配 ENI 后，任务将注册到 AWS 安全组中。安全组充当 VPC 的虚拟防火墙，您可以使用该防火墙控制入站和出站流量。

AWS PrivateLink 和 Amazon ECS

AWS PrivateLink 是一种网络技术，使您能够为包括 Amazon ECS 在内的不同 AWS 服务创建私有端点。在没有连接到 Amazon VPC 的互联网网关（IGW），也没有通往互联网的备用路由的沙盒环境中，需要端点。使用 AWS PrivateLink 可确保对 Amazon ECS 服务的调用保持在 Amazon VPC 内且不会产生互联网流量。有关如何为 Amazon ECS 和其他相关服务创建 AWS PrivateLink 端点的说明，请参阅 Amazon ECS 接口 Amazon VPC 端点。

> **Important**
>
> AWS Fargate 任务不需要使用 Amazon ECS 的 AWS PrivateLink 端点。

Amazon ECR 和 Amazon ECS 都支持端点策略。这些策略使您能够完善对服务 API 的访问权限。例如，您可以为 Amazon ECR 创建端点策略，该策略仅允许将映像推送到特定 AWS 账户的注册表。这样的策略可以用来防止数据通过容器映像泄露，同时仍然允许用户推送到授权的 Amazon ECR 注册表。有关更多信息，请参阅使用 VPC 端点策略。

以下策略仅允许您账户中的所有 AWS 主体对您的 Amazon ECR 存储库执行所有操作：

```
{
  "Statement": [
    {
      "Sid": "LimitECRAccess",
      "Principal": "*",
      "Action": "*",
      "Effect": "Allow",
      "Resource": "arn:aws:ecr:region:account_id:repository/*"
    },
  ]
}
```

您可以通过设置使用新 PrincipalOrgID 属性的条件来进一步增强这一点。这样可以防止不属于您的 AWS Organizations 的 IAM 主体推送和拉取映像。有关更多信息，请参阅 aws:PrincipalOrgID。

建议对 com.amazonaws.*region*.ecr.dkr 和 com.amazonaws.*region*.ecr.api 端点应用相同的策略。

容器代理设置

Amazon ECS 容器代理配置文件包含多个与网络安全相关的环境变量。ECS_AWSVPC_BLOCK_IMDS 和 ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST 用于阻止任务访问 Amazon EC2 元数据。HTTP_PROXY 用于将代理配置为通过 HTTP 代理进行路由以连接到互联网。有关将代理和 Docker 运行时系统配置为通过代理进行路由的说明，请参阅 HTTP 代理配置。

> **Important**
>
> 当您使用 AWS Fargate 时，这些设置不可用。

网络安全建议

建议您在设置 Amazon VPC、负载均衡器和网络时执行以下操作。

在适用的情况下将网络加密与 Amazon ECS 结合使用

您应该在适用的情况下使用网络加密。某些合规计划（例如 PCI DSS）要求您对传输中的数据进行加密，前提是这些数据包含持卡人数据。如果您的工作负载有类似的要求，请配置网络加密。

当用户连接到不安全的网站时，现代浏览器会发出警告。如果您的服务前面有面向公众的负载均衡器，请使用 TLS/SSL 加密从客户端浏览器到负载均衡器的流量，并在必要时重新加密到后端。

使用 **awsvpc** 网络模式和安全组控制任务与 Amazon ECS 中的其他资源之间的流量

当您需要控制任务之间和任务与其他网络资源之间的流量时，您应该使用 awsvpc 网络模式和安全组。如果您的服务位于 ALB 之后，则使用安全组仅允许来自其他网络资源的入站流量，这些资源使用与您的 ALB 相同的安全组。如果您的应用程序位于 NLB 之后，请将任务的安全组配置为仅允许来自 Amazon VPC CIDR 范围的入站流量以及分配给 NLB 的静态 IP 地址。

还应使用安全组来控制任务与 Amazon VPC 内其他资源（例如 Amazon RDS 数据库）之间的流量。

当需要严格隔离网络流量时，在单独的 Amazon VPC 中创建 Amazon ECS 集群

当需要严格隔离网络流量时，您需要在单独的 Amazon VPC 中创建集群。避免在集群上使用不必遵守严格安全要求的工作负载运行具有严格安全要求的工作负载。当必须进行严格的网络隔离时，请在单独

的 Amazon VPC 中创建集群，并有选择地使用 Amazon VPC 端点向其他 Amazon VPC 公开服务。有关更多信息，请参阅 Amazon VPC 端点。

在必要时为 Amazon ECS 配置 AWS PrivateLink 端点

您应在必要时配置 AWS PrivateLink 端点。如果您的安全策略防止您将互联网网关（IGW）连接到 Amazon VPC，请为 Amazon ECS 以及 Amazon ECR、AWS Secrets Manager 和 Amazon CloudWatch 等其他服务配置 AWS PrivateLink 端点。

使用 Amazon VPC 流日志分析 Amazon ECS 中的长时间运行任务的进出流量

您应该使用 Amazon VPC 流日志分析长时间运行的任务的进出流量。使用 awsvpc 网络模式的任务会获得自己的 ENI。为此，您可以使用 Amazon VPC 流日志监控进出单个任务的流量。Amazon VPC 流日志（v3）的最新更新使用流量元数据丰富了日志，包括 vpc ID、子网 ID 和实例 ID。此元数据可用于帮助缩小调查范围。有关更多信息，请参阅 Amazon VPC 流日志。

> **Note**
>
> 由于容器的临时性质，流日志可能并不总是分析不同容器或容器与其他网络资源之间的流量模式的有效方法。

EC2 启动类型的 Amazon ECS 任务联网选项

Amazon EC2 实例上托管的 Amazon ECS 任务的联网行为取决于任务定义中所定义的网络模式。我们建议您使用 awsvpc 网络模式，除非您特别需要使用其他网络模式。

以下是可用的网络模式。

网络模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	Description
awsvpc	是	是	向任务分配其自己的弹性网络接口（ENI）和主要私有 IPv4 地址。这将为任务提供与 Amazon EC2 实例相同的网络属性。
bridge	是	不支持	任务使用 Linux 上的 Docker 内置虚拟网络，该网络在托管任务的每个 Amazon EC2 实例内运行。Linux 上的内置虚拟网络使用 bridge

网络模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	Description
			Docker 网络驱动程序。如果未在任务定义中指定网络模式，则这是 Linux 上的默认网络模式。
`host`	是	不支持	任务通过直接映射容器端口到托管任务的 Amazon EC2 实例的 ENI，使用会绕过 Docker 内置虚拟网络的主机网络。动态端口映射不可在此网络模式中使用。使用此模式的任务定义中的容器必须指定具体的 `hostPort` 号。主机上的端口号不可被用于多个任务。因此，您不可以在单个 Amazon EC2 实例上运行相同任务定义的多个任务。
`none`	是	不支持	任务没有外部网络连接。
`default`	否	是	任务使用 Windows 上的 Docker 内置虚拟网络，该网络在托管任务的每个 Amazon EC2 实例内运行。Windows 上的内置虚拟网络使用 `nat` Docker 网络驱动程序。如果未在任务定义中指定网络模式，则这是 Windows 上的默认网络模式。

有关 Linux 上的 Docker 联网的更多信息，请参阅 Docker 文档中的联网概述。

有关 Windows 上的 Docker 联网的更多信息，请参阅 Windows 上的 Microsoft 容器文档中的 Windows 容器联网。

为 Amazon ECS 任务分配网络接口

`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。使用 `awsvpc` 网络模式可简化容器联网，因为您可以更好地控制您的应用程序相互之间以及与 VPC 内其他服务之间的通信方式。`awsvpc` 网络模式还为您的容器提供了更高的安全性，使您能够在任务中更精细地使用安全组和网络监视工具。您也可以利用其他 Amazon EC2 联网功能（例如 VPC 流日志），以便监控来往于您的任务的流量。此外，属于同一任务的容器可以通过 `localhost` 接口进行通信。

任务弹性网络接口（ENI）是 Amazon ECS 的完全托管功能。Amazon ECS 创建 ENI 并将它附加到具有指定安全组的 Amazon EC2 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。预设情况下，每个任务 ENI 都会分配一个专用 IPv4 地址。如果您的 VPC 启用了双堆栈模式，并且您使用带有 IPv6 CIDR 块的子网，则任务 ENI 也将收到 IPv6 地址。每个任务只能有一个 ENI。

这些 ENI 在您账户的 Amazon EC2 控制台中可见。您的账户无法分离或修改 ENI。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。

当您需要提高的 ENI 密度时，请使用 `awsvpcTrunking` 账户设置。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 托管。在您从 Amazon ECS 集群中终止或注销容器实例时，将删除中继 ENI。有 `awsvpcTrunking` 账户设置的更多信息，请参阅 先决条件。

您可以在任务定义的 `networkMode` 参数中指定 `awsvpc`。有关更多信息，请参阅 网络模式。

然后在运行任务或创建服务时使用 `networkConfiguration` 参数，其中包含用于将任务放置到一个或多个安全组的一个或多个子网，以附加到某个 ENI。有关更多信息，请参阅 网络配置。这些任务放置在与这些子网相同的可用性区域中的兼容 Amazon EC2 实例上，并且指定的安全组与为该任务预置的 ENI 相关联。

Linux 注意事项

在使用 Linux 操作系统时考虑以下事项。

- 如果您在 `awsvpc` 模式下使用 a p5.48xlarge 实例，则在该实例上运行的任务不能超过 1 个。
- 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 Management Console 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。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
  ```

- 您的 Amazon EC2 Linux 实例需要版本 `1.15.0` 或更高版本的容器代理运行使用 `awsvpc` 网络模式。如果您正在使用经 Amazon ECS 优化的 AMI，您的实例将需要不低于 `1.15.0-4` 版本的 `ecs-init` 程序包。
- 在 VPC 上启用 `enableDnsHostnames` 和 `enableDnsSupport` 选项时， Amazon ECS使用任务联网和 Amazon 提供的（内部）DNS 主机名来填充任务的主机名。如果未启用这些选项，则任务的

DNS 主机名将被设置为随机主机名。有关 VPC 的 DNS 设置的更多信息，请参阅《Amazon VPC 用户指南》中的在 VPC 中使用 DNS。

- 每个使用 awsvpc 网络模式的 Amazon ECS 任务都会接收到自己的 elastic network interface (ENI)，该网络接口附加到托管它的 Amazon EC2 实例。可以附加到 Amazon EC2 Linux 实例的网络接口数量有默认配额。主网络接口算作该配额中的一个。例如，默认情况下，最多只能将三个 ENI 附加到一个 c5.large 实例。实例的主网络接口计为一个。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。要详细了解每种实例类型的默认 ENI 限制，请参阅《Amazon EC2 用户指南》中的每种实例类型的每个网络接口的 IP 地址数。
- Amazon ECS 支持使用支持的实例类型启动 Amazon EC2 Linux 实例，并增加 ENI 密度。当您选择加入 awsvpcTrunking 账户设置并将使用这些实例类型的 Amazon EC2 Linux 实例注册到集群时，这些实例具有更高的 ENI 配额。将这些实例与此更高配额结合使用意味着您可以在每个 Amazon EC2 Linux 实例上放置更多的任务。为了将增加的 ENI 密度与中继功能结合使用，您的 Amazon EC2 实例必须使用版本 1.28.1 或更高版本的容器代理。如果您正在使用经 Amazon ECS 优化的 AMI，您的实例将还需要不低于版本 1.28.1-2 的 ecs-init 软件包。有关选择使用 awsvpcTrunking 账户设置的更多信息，请参阅 通过账户设置访问 Amazon ECS 功能。有关 ENI 中继的更多信息，请参阅增加 Amazon ECS Linux 容器实例网络接口。
- 当在 Amazon EC2 Linux 实例上托管使用 awsvpc 网络模式的任务时，不会为您的任务 ENI 提供公有 IP 地址。要访问互联网，必须在配置为使用 NAT 网关的私有子网中启动任务。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT 网关。入站网络访问必须从使用私有 IP 地址的 VPC 内进行，或者通过 VPC 内的负载均衡器进行路由。在公有子网中启动的任务无法访问互联网。
- Amazon ECS 仅为您识别连接到 Amazon EC2 Linux 实例的 ENI。如果您手动将 ENI 附加到您的实例，Amazon ECS 可能会尝试向没有足够网络适配器的实例添加任务。这可能会导致任务超时并转为取消预置状态，然后变为已停止状态。我们建议您不要手动将 ENI 附加到实例。
- Amazon EC2 Linux 实例必须注册到可以考虑在 awsvpc 网络模式下放置任务的 ecs.capability.task-eni 功能。运行版本 1.15.0-4 或更高版本的 ecs-init 的实例注册自动有此属性。
- 创建并连接到 Amazon EC2 Linux 实例的 ENI 不能手动分离或由您的账户修改。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 参考中的 AwsVpcConfiguration。
- 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，它通过运行 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。

- 任务使用 awsvpc 网络模式的服务只支持应用程序负载均衡器和网络负载均衡器。当您为这些服务创建任何目标组时，必须选择 ip 作为目标类型。切勿使用 instance。这是因为使用 awsvpc 网络模式的任务与 ENI 而不是 Amazon EC2 Linux 实例关联。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。
- 如果您的 VPC 进行了更新以更改其使用的 DHCP 选项集，则无法将这些更改应用于现有任务。在应用这些更改的情况下启动新任务，验证它们是否正常工作，然后停止现有任务以安全地更改这些网络配置。

Windows 注意事项

以下是使用 Windows 操作系统时的注意事项：

- 使用经 Amazon ECS 优化的 Windows Server 2016 AMI 的容器实例无法托管使用 awsvpc 网络模式的任务。如果您的集群包含经 Amazon ECS 优化的 Windows Server 2016 AMI 和支持 awsvpc 网络模式的 Windows AMI，则使用 awsvpc 网络模式的任务不会在 Windows 2016 Server 实例上启动。而是会在支持 awsvpc 网络模式的实例上启动。
- 您的 Amazon EC2 Windows 实例需要版本 1.57.1 或更高版本的容器代理为使用 awsvpc 网络模式的 Windows 容器使用 CloudWatch 指标。
- 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 Management Console 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。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 Amazon EC2 Windows 实例需要版本 1.54.0 或更高版本的容器代理运行使用 awsvpc 网络模式。在引导实例时，必须配置 awsvpc 网络模式所需的选项。有关更多信息，请参阅 the section called “引导启动容器实例”。
- 在 VPC 上同时启用 enableDnsHostnames 和 enableDnsSupport 选项时，Amazon ECS 会使用 Amazon 提供的（内部）DNS 主机名来填充任何主机名。如果未启用这些选项，则任务的 DNS 主机名为随机主机名。有关 VPC 的 DNS 设置的更多信息，请参阅《Amazon VPC 用户指南》中的在 VPC 中使用 DNS。
- 每个使用 awsvpc 网络模式的 Amazon ECS 任务都会接收到自己的 elastic network interface (ENI)，该网络接口附加到托管它的 Amazon EC2 Windows 实例。可以附加到 Amazon EC2

Windows 实例的网络接口数量有默认配额。主网络接口算作该配额中的一个。例如，默认情况下，最多只可将三个 ENI 附加到一个 c5.large 实例。实例的主网络接口计为其中一个。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。要详细了解每种实例类型的默认 ENI 限制，请参阅《Amazon EC2 用户指南》中的每种实例类型的每个网络接口的 IP 地址数。

- 当在 Amazon EC2 Windows 上托管使用 awsvpc 网络模式时，不会为您的任务 ENI 提供公有 IP 地址。要访问互联网，请在配置为使用 NAT 网关的私有子网中启动任务。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 地址从 VPC 内进行，或者通过 VPC 内的负载均衡器进行路由。在公有子网中启动的任务无法访问互联网。
- Amazon ECS 仅为您识别已附加到 Amazon EC2 Windows 实例的 ENI。如果您手动将 ENI 附加到您的实例，Amazon ECS 可能会尝试向没有足够网络适配器的实例添加任务。这可能会导致任务超时并转为取消预置状态，然后变为已停止状态。我们建议您不要手动将 ENI 附加到实例。
- Amazon EC2 Windows 实例必须注册到可以考虑在 awsvpc 网络模式下放置任务的 ecs.capability.task-eni 功能。
- 您不能手动修改或分离创建并连接到 Amazon EC2 Windows 实例的 ENI。这是为了防止您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。
- 当运行任务或创建使用 awsvpc 网络模式的服务时，您最多只能在 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 参考中的 AwsVpcConfiguration。
- 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，它通过运行 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。
- 任务使用 awsvpc 网络模式的服务只支持应用程序负载均衡器和网络负载均衡器。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 Linux Windows 关联。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。
- 如果您的 VPC 进行了更新以更改其使用的 DHCP 选项集，则无法将这些更改应用于现有任务。在应用这些更改的情况下启动新任务，验证它们是否正常工作，然后停止现有任务以安全地更改这些网络配置。
- 在 EC2 Windows 配置中使用 awsvpc 网络模式时，不支持以下内容：
  - 双堆栈配置
  - IPv6

- ENI 中继

在双堆栈模式下使用 VPC

在双堆栈模式下使用 VPC 时，您的任务可通过 IPv4 或 IPv6 或两者进行通信。IPv4 和 IPv6 地址彼此独立。因此，您必须在 VPC 中分别针对 IPv4 和 IPv6 配置路由和安全设置。有关如何将 VPC 配置为双堆栈模式的更多信息，请参阅《Amazon VPC 用户指南》 中的迁移到 IPv6。

如果您使用互联网网关或仅出站互联网网关配置 VPC，则可以在双堆栈模式下使用 VPC。通过这样做，分配了 IPv6 地址的任务可以通过互联网网关或仅限出口的互联网网关访问互联网。NAT 网关是可选项。有关更多信息，请参阅 Amazon VPC 用户用户指南中的互联网网关和 Egress-only 互联网网关。

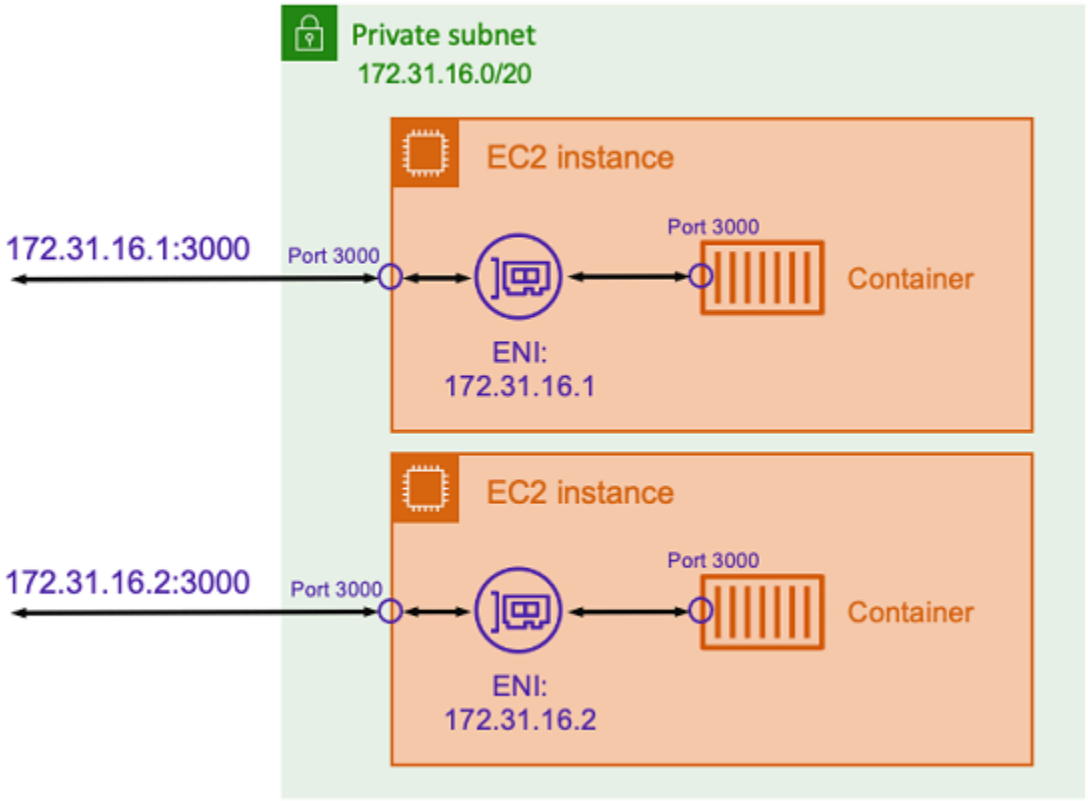
如果满足以下条件，则向 Amazon ECS 任务分配 IPv6 地址：

- 托管该任务的 Amazon EC2 Linux 实例正在使用版本 `1.45.0` 或更高版本的容器代理。有关如何检查您的实例正在使用的代理版本以及根据需要进行更新的信息，请参阅更新 Amazon ECS 容器代理。
- `dualStackIPv6` 账户设置为已启用。有关更多信息，请参阅 通过账户设置访问 Amazon ECS 功能。
- 您的任务是使用 `awsvpc` 网络模式。
- 您的 VPC 和子网已为 IPv6 配置。配置包括在指定子网中创建的网络接口。有关如何将 VPC 配置为双堆栈模式的更多信息，请参阅《Amazon VPC 用户指南》中的迁移到 IPv6 和为您的子网修改 IPv6 寻址属性。

将 Amazon ECS 容器端口映射到 EC2 实例网络接口

`host` 网络模式仅支持 Amazon EC2 实例上托管的 Amazon ECS 任务。在 Fargate 上使用 Amazon ECS 时不支持该功能。

`host` 网络模式是 Amazon ECS 支持的最基本的网络模式。使用主机模式时，容器的网络连接直接绑定到运行容器的底层主机。



假设您正在运行一个 Node.js 容器，其中包含一个 Express 应用程序，该应用程序监听类似于上图所示端口的端口 3000。使用 host 网络模式时，容器使用底层主机 Amazon EC2 实例的 IP 地址在端口 3000 上接收流量。不建议使用此模式。

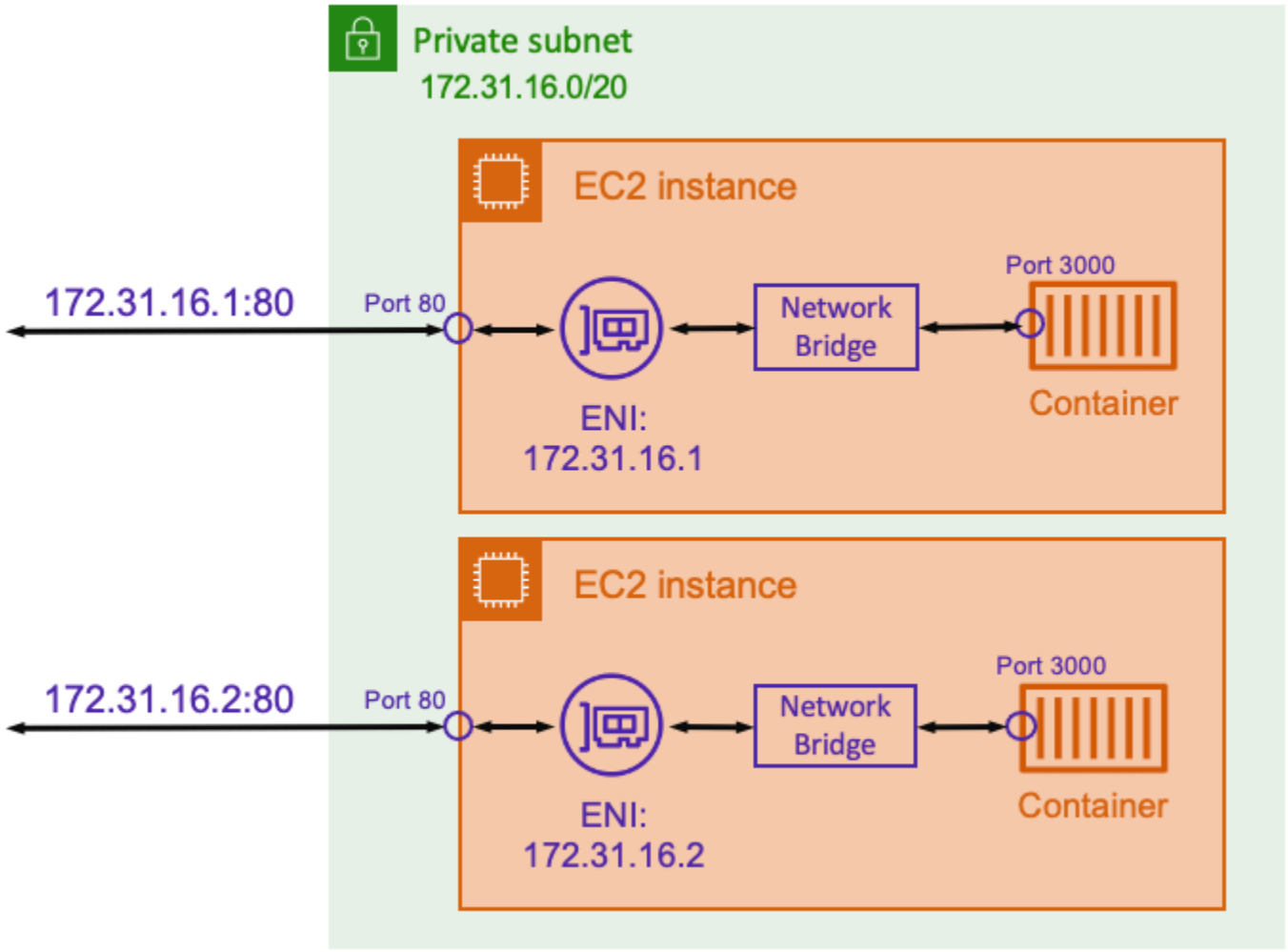
使用这种网络模式有明显的缺点。在每台主机上只能运行一个任务的单个实例化。这是因为只有第一个任务可以绑定到 Amazon EC2 实例上所需的端口。使用 host 网络模式时，也无法重新映射容器端口。例如，如果应用程序需要侦听特定的端口号，则您无法直接重新映射该端口号。相反，您必须通过更改应用程序配置来管理所有端口冲突。

使用 host 网络模式还存在安全影响。此模式允许容器模拟主机，并允许容器连接到主机上的私有环回网络服务。

将 Docker 的虚拟网络用于 Amazon ECS Linux 任务

bridge 网络模式仅支持 Amazon EC2 实例上托管的 Amazon ECS 任务。

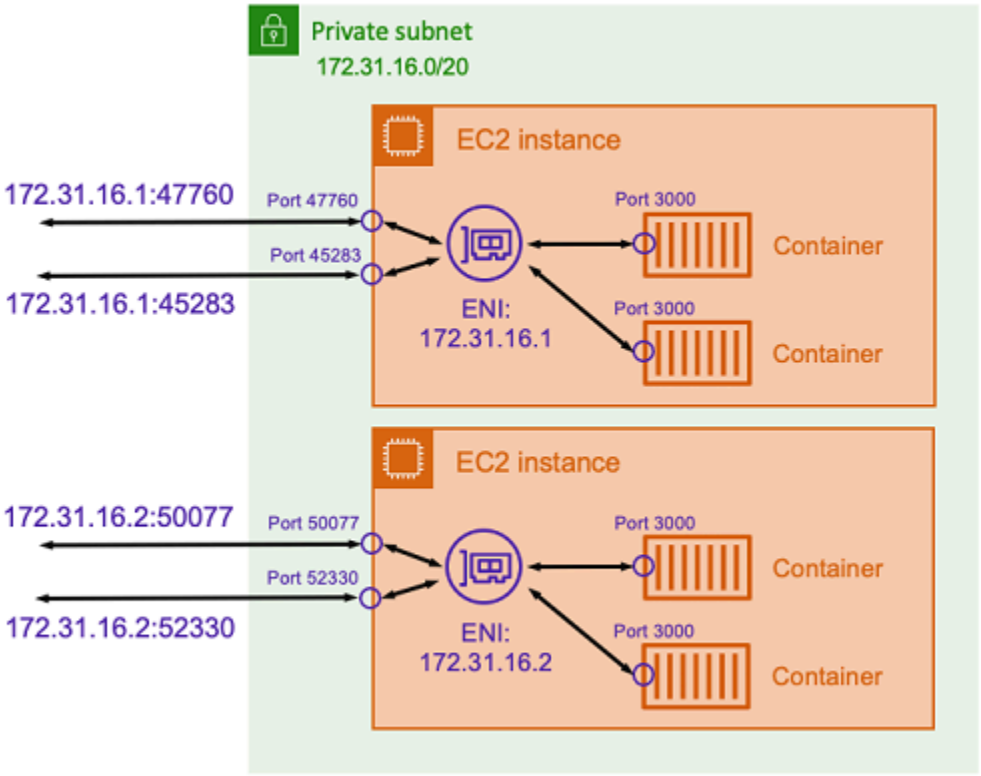
在 bridge 模式下，您使用虚拟网桥在主机和容器的网络之间创建一个层。这样，您就可以创建将主机端口重新映射到容器端口的端口映射。映射可以是静态，也可以是动态。



使用静态端口映射，您可以明确定义要映射到容器端口的主机端口。使用上面的示例，主机上的端口 80 被映射到容器上的端口 3000。要与容器化应用程序通信，您需要将流量发送到 Amazon EC2 实例的 IP 地址的端口 80。从容器化应用程序的角度来看，它可以看到端口 3000 上的入站流量。

如果您只想更改流量端口，则静态端口映射是合适的。但是，这仍然具有与使用 host 网络模式相同的缺点。在每台主机上只能运行一个任务的单个实例化。这是因为静态端口映射仅允许将单个容器映射到端口 80。

要解决此问题，请考虑使用带有动态端口映射的 bridge 网络模式，如下图所示。



通过不在端口映射中指定主机端口，您可以让 Docker 从临时端口范围中选择随机的、未使用的端口，并将其分配为容器的公共主机端口。例如，监听容器上端口 3000 的 Node.js 应用程序可能会被分配一个随机的大数值端口，例如 Amazon EC2 主机上的 47760。这样做意味着您可以在主机上运行该容器的多个副本。此外，还可以在主机上为每个容器分配自己的端口。容器的每个副本都会在端口 3000 上接收流量。但是，向这些容器发送流量的客户端使用随机分配的主机端口。

Amazon ECS 可帮助您跟踪为每项任务随机分配的端口。它通过自动更新负载均衡器目标组和 AWS Cloud Map 服务发现以包含任务 IP 地址和端口列表来实现此目的。这样可以更轻松地使用通过动态端口使用 bridge 模式运行的服务。

但是，使用 bridge 网络模式的一项缺点是，很难将服务锁定为服务通信。由于服务可能会分配给任何随机的、未使用的端口，因此有必要在主机之间开放广泛的端口范围。但是，要创建特定的规则，使某项特定服务只能与另一项特定服务进行通信并不容易。这些服务没有用于安全组网络规则的特定端口。

Fargate 启动类型的 Amazon ECS 任务联网选项

预设情况下，Fargate 上的每个 Amazon ECS 任务都会提供 elastic network interface (ENI) 和主要私有 IP 地址。使用公有子网时，您可以选择向任务的 ENI 分配公有 IP 地址。如果您的 VPC 配置了双堆

栈模式，并且您使用具有 IPv6 CIDR 块的子网，则您的任务 ENI 也将接收 IPv6 地址。一个任务在给定的时间只能有一个与之关联的 ENI。属于同一任务的容器可以通过 `localhost` 接口进行通信。有关 VPC 和子网的更多信息，请参阅 Amazon VPC 用户指南中的 VPC 和子网。

要使 Fargate 上的任务拉取容器映像，任务必须具有通往互联网的路由。下面说明如何验证您的任务具有通往互联网的路由。

- 使用公有子网时，您可以向任务 ENI 分配公有 IP 地址。
- 使用私有子网时，子网可以连接一个 NAT 网关。
- 使用 Amazon ECR 中托管的容器映像时，您可以将 Amazon ECR 配置为使用接口 VPC 端点，将通过任务的私有 IPv4 地址进行映像提取 。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。

由于每个任务都有自己的 ENI，因此您可以使用联网功能（例如 VPC 流日志），以便可以监控您任务的流量。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 流日志。

您还可以利用 AWS PrivateLink。您可以配置 VPC 接口端点，以便可以通过私有 IP 地址访问 Amazon ECS API。AWS PrivateLink 将 VPC 和 Amazon ECS 之间的所有网络流量限制在 Amazon 网络以内。您无需互联网网关、NAT 设备或虚拟私有网关。有关更多信息，请参阅 Amazon ECS 最佳实践指南中的 AWS PrivateLink。

有关如何将 `NetworkConfiguration` 资源与 AWS CloudFormation 一起使用的示例，请参阅 the section called “使用单独的堆栈创建 Amazon ECS 资源”。

创建的 ENI 由 AWS Fargate 完全托管。此外，还有一个关联的 IAM policy 用于向 Fargate 授予权限。对于使用 Fargate 平台版本 `1.4.0` 或更高版本的任务，任务会接收单个 ENI（称为任务 ENI），所有网络流量都将流经 VPC 内的这个 ENI。此流量记录在您的 VPC 流日志中。对于使用 Fargate 平台版本 `1.3.0` 及更早版本的任务，除了任务 ENI 外，任务还会收到单独的 Fargate 拥有的 ENI，该 ENI 用于某些在 VPC 流日志中不可见的网络流量。下表介绍网络流量行为以及每个平台版本所需的 IAM policy。

操作	Linux 平台版本 **`1.3.0`** 及更早版本的流量	Linux 平台版本 **`1.4.0`** 的流量	Windows 平台版本 **`1.0.0`** 的流量	IAM 权限
检索 Amazon ECR 登录凭据	Fargate 拥有的 ENI	任务 ENI	任务 ENI	任务执行 IAM 角色

操作	Linux 平台版本 **1.3.0** 及更早版本的流量	Linux 平台版本 **1.4.0** 的流量	Windows 平台版本 **1.0.0** 的流量	IAM 权限
镜像提取	任务 ENI	任务 ENI	任务 ENI	任务执行 IAM 角色
通过日志驱动程序发送日志	任务 ENI	任务 ENI	任务 ENI	任务执行 IAM 角色
通过适用于 Amazon ECS 的 FireLens 发送日志	任务 ENI	任务 ENI	任务 ENI	任务 IAM 角色
从 Secrets Manager 或 Systems Manager 中检索密码	Fargate 拥有的 ENI	任务 ENI	任务 ENI	任务执行 IAM 角色
Amazon EFS 文件系统流量	不可用	任务 ENI	任务 ENI	任务 IAM 角色
应用程序流量	任务 ENI	任务 ENI	任务 ENI	任务 IAM 角色

注意事项

在使用任务联网时考虑以下事项。

- Amazon ECS 服务链接角色需要为 Amazon ECS 提供代表您呼叫其他 AWS 服务的权限。此角色是在创建集群时或者在 AWS Management Console 中创建或更新服务时为您创建的。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。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
```

- 在 VPC 上启用 `enableDnsHostnames` 和 `enableDnsSupport` 选项时，Amazon ECS 使用 Amazon 提供的 DNS 主机名来填充任务的主机名。如果未启用这些选项，则任务的 DNS 主机名将

被设置为随机主机名。有关 VPC 的 DNS 设置的更多信息，请参阅《Amazon VPC 用户指南》中的在 VPC 中使用 DNS。

- 您最多只能为 `awsVpcConfiguration` 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 参考中的 AwsVpcConfiguration。
- 您不能手动分离或修改 Fargate 创建并附加的 ENI。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。
- 如果 VPC 子网进行了更新以更改其使用的 DHCP 选项集，则无法将这些更改应用于使用 VPC 的现有任务。启动新任务，以在测试新更改的同时获得平稳迁移的新设置，然后在不需要回滚的情况下停止旧的任务。
- 使用 Fargate 平台版本 `1.4.0` 或更高版本（对于 Linux）或者 `1.0.0`（对于 Windows）时，在具有 IPv6 CIDR 块的子网中启动的任务仅接收 IPv6 地址。
- 对于使用平台版本 `1.4.0` 或更高版本（对于 Linux）或 `1.0.0`（对于 Windows）的任务，任务 ENI 支持巨型帧。网络接口配置了最大传输单元 (MTU)，这是单个帧内将放入的最大负载的大小。MTU 越大，单个帧内可以放入的应用程序负载就越多，这可以减少每帧开销并提高效率。当您的任务和目标之间的网络路径支持巨型帧时，支持巨型帧将减少开销。
- 任务使用 Fargate 启动类型的服务仅支持应用程序负载均衡器和网络负载均衡器。不支持经典负载均衡器。当您创建任何目标组时，必须选择 `ip` 而不是 `instance` 作为目标类型。有关更多信息，请参阅 使用负载均衡分配 Amazon ECS 服务流量。

在双堆栈模式下使用 VPC

在双堆栈模式下使用 VPC 时，您的任务可通过 IPv4 /或 IPv6 或两者进行通信。IPv4 和 IPv6 地址是彼此独立的；您必须在 VPC 中分别针对 IPv4 和 IPv6 配置路由和安全设置。有关将 VPC 配置为双堆栈模式的更多信息，请参阅 Amazon VPC用户指南 中的迁移到 IPv6。

如果满足以下条件，将向 Fargate 上的 Amazon ECS 任务分配 IPv6 地址：

- 您的 Amazon ECS `dualStackIPv6` 账户设置为 IAM 主体在您启动任务所在的区域启动您的任务时打开（`enabled`）。此设置仅可使用 API 或 AWS CLI 修改。您可以选择为账户中的特定 IAM 主体开启此设置，也可以通过设置账户默认设置为整个账户开启此设置。有关更多信息，请参阅 通过账户设置访问 Amazon ECS 功能。
- 您的 VPC 和子网已为 IPv6 启用。有关如何将 VPC 配置为双堆栈模式的更多信息，请参阅《Amazon VPC 用户指南》 中的迁移到 IPv6。
- 您的子网已启用自动分配 IPv6 地址。有关如何配置子网的更多信息，请参阅《Amazon VPC 用户指南》中的修改子网的 IPv6 寻址属性。

- 任务或服务使用 Fargate 平台版本 `1.4.0` 或更高版本（用于 Linux）。

如果您使用互联网网关或仅限出现的互联网网关配置 VPC，分配了 IPv6 地址的 Fargate 上的 Amazon ECS 任务可以访问互联网。不需要 NAT 网关。有关更多信息，请参阅 Amazon VPC 用户用户指南中的互联网网关和 Egress-only 互联网网关。

Amazon ECS 任务的存储选项

Amazon EC2 为您提供了灵活、经济高效且易于使用的数据存储选项，具体取决于您的需求。Amazon ECS 支持以下容器的数据卷选项：

数据量	支持的启动类型	支持的操作系统	存储持久性	使用案例
Amazon Elastic Block Store（Amazon EBS)	Fargate、Amazon EC2	Linux	附加到独立任务时可以持久保存。附加到服务维护的任务时是短暂的。	Amazon EBS 卷为数据密集型容器化工作负载提供经济高效、持久、高性能的块存储。常见使用情况包括事务性工作负载（例如数据库、虚拟桌面和根卷）以及吞吐量密集型工作负载（例如日志处理和 ETL 工作负载）。有关更多信息，请参阅 将 Amazon EBS 卷与 Amazon ECS 结合使用。
Amazon Elastic File System (Amazon EFS)	Fargate、Amazon EC2	Linux	持续的	Amazon EFS 卷提供简单、可扩展和持久的共享文件存储用于

数据量	支持的启动类型	支持的操作系统	存储持久性	使用案例
				Amazon ECS 任务，它会随着您添加和删除文件而自动增长和缩小。Amazon EFS 卷支持并发性，对于水平扩展且需要低延迟、高吞吐量和先写后读一致性等存储功能的容器化应用程序，非常有用。常见使用案例包括数据分析、媒体处理、内容管理和 Web 服务等工作负载。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。

数据量	支持的启动类型	支持的操作系统	存储持久性	使用案例
适用于 Windows File Server 的 Amazon FSx	Amazon EC2	Windows	持续的	FSx for Windows File Server 卷提供完全托管的 Windows 文件服务器，您可以使用这些服务器来预调配需要持久、分布式、共享和静态文件存储的 Windows 任务。常见使用案例包括 .NET 应用程序，这些应用程序可能需要本地文件夹作为持久存储来保存应用程序输出。适用于 Windows File Server 的 Amazon FSx 在容器中提供了本地文件夹，使多个容器能够在由 SMB 共享支持的同一个文件系统上进行读写操作。有关更多信息，请参阅 将 FSx for Windows File Server 卷与 Amazon ECS 结合使用。

数据量	支持的启动类型	支持的操作系统	存储持久性	使用案例
Docker 卷	Amazon EC2	Windows、Linux	持续的	Docker 卷是 Docker 容器运行时的一项功能，它允许容器通过从主机的文件系统挂载目录来持久保存数据。Docker 卷驱动程序（也称为插件）用于将容器卷与外部存储系统集成。Docker 卷可以由第三方驱动程序或内置的 `local` 驱动程序管理。Docker 卷的常见使用案例包括提供持久数据卷或在同一容器实例的不同容器上的不同位置共享卷。有关更多信息，请参阅将 Docker 卷与 Amazon ECS 结合使用。

数据量	支持的启动类型	支持的操作系统	存储持久性	使用案例
绑定挂载	Fargate、Amazon EC2	Windows、Linux	临时的	绑定挂载由挂载到容器中的主机上的文件或目录组成，例如 Amazon EC2 实例或 AWS Fargate。绑定挂载的常见使用案例包括与同一任务中的其他容器共享源容器中的卷，或者在一个或多个容器中挂载主机卷或空卷。有关更多信息，请参阅 将绑定挂载与 Amazon ECS 结合使用。

将 Amazon EBS 卷与 Amazon ECS 结合使用

Amazon Elastic Block Store（Amazon EBS）卷为数据密集型工作负载提供高度可用、经济高效、持久、高性能的块存储。Amazon EBS 卷可以与 Amazon ECS 任务一起用于高吞吐量和事务密集型应用程序。

在独立任务启动期间，您可以提供用于将一个 EBS 卷附加到任务的配置。在创建或更新服务期间，您可以提供用于将每个任务的一个 EBS 卷附加到 ECS 服务管理的每个任务的配置。

通过在启动时提供卷配置而不是在任务定义中提供，您可以创建不受特定数据卷类型或特定 EBS 卷设置限制的任务定义。然后，您可以在不同的运行时环境中重复使用任务定义。例如，在部署期间，您可以为生产工作负载提供比预生产环境更多的吞吐量。

附加到 Amazon ECS 任务的 Amazon EBS 卷由 Amazon ECS 代表您管理。可以使用 AWS Key Management Service（AWS KMS）密钥对卷进行加密以保护您的数据。您可以配置新的空卷以进行附加，也可以使用快照从现有卷加载数据。

要监控卷的性能，您还可以使用 Amazon CloudWatch 指标。有关 Amazon EBS 卷的 Amazon ECS 指标的更多信息，请参阅 Amazon ECS CloudWatch 指标 和 Amazon ECS Container Insights 指标。

有关 Amazon EBS 卷的更多信息，请参阅《Amazon EBS 用户指南》https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volumes.html中的 Amazon EBS 卷。

Amazon EBS 卷的 AWS 区域 和可用区

Amazon EBS 卷可以在以下 AWS 区域 中附加到 Amazon ECS 任务：

区域名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国东部（俄亥俄州）	us-east-2
美国西部（北加利福尼亚）	us-west-1
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（墨尔本）	ap-southeast-4
亚太地区（孟买）	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
亚太地区（新加坡）	ap-southeast-1

区域名称	区域代码
亚太地区（悉尼）	ap-southeast-2
亚太地区（东京)	ap-northeast-1
加拿大（中部）	ca-central-1
欧洲地区（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（米兰）	eu-south-1
欧洲地区（巴黎）	eu-west-3
欧洲（西班牙）	eu-south-2
欧洲地区（斯德哥尔摩）	eu-north-1
欧洲（苏黎世）	eu-central-2
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南美洲（圣保罗）	sa-east-1

Important

您无法将 Amazon EBS 卷配置为附加到 `euc1-az2` 和 `use1-az3` 可用区中的 Fargate Amazon ECS 任务。

注意事项

使用 Amazon EBS 卷时应考虑以下事项：

- Amazon EBS 卷仅支持托管在 Fargate 上的 Linux 任务，以及托管在基于 Nitro Linux 实例上具有经 Amazon ECS 优化的亚马逊机器映像（AMI）的 EC2 启动类型任务。有关实例类型的信息，请参阅《Amazon EC2 用户指南》中的实例类型。有关 Amazon ECS 启动类型的更多信息，请参阅Amazon ECS 启动类型。
- 对于在 Fargate 上托管的任务，平台版本 `1.4.0` 或更高版本（Linux）支持 Amazon EBS 卷。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。
- 对于托管在 Amazon EC2 Linux 实例上的任务，经 ECS 优化的 `20231219` AMI 或更高版本支持 Amazon EBS 卷。有关更多信息，请参阅检索经 Amazon ECS 优化的 AMI 元数据。
- 托管在 Fargate 上的任务不支持磁性（`standard`）Amazon EBS 卷类型。有关 Amazon EBS 卷类型的更多信息，请参阅《Amazon EC2 用户指南》https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html中的 Amazon EBS 卷。
- 创建服务或在部署时配置卷的独立任务时，需要 Amazon ECS 基础设施的 IAM 角色。您可以将 AWS 托管的 `AmazonECSInfrastructureRolePolicyForVolumes` IAM 策略附加到角色，也可以使用托管策略作为指南，以创建并附加您自己的具有满足您特定需求的权限的策略。有关更多信息，请参阅 Amazon ECS 基础设施 IAM 角色。
- 您最多可以将一个 Amazon EBS 卷附加到每个 Amazon ECS 任务，并且该卷必须是新卷。您不能将现有的 Amazon EBS 卷挂载到任务中。但是，您可以在部署时使用现有卷的快照来配置新的 Amazon EBS 卷。
- 您只能在部署时为使用滚动更新部署类型和副本计划策略的服务配置 Amazon EBS 卷。
- Amazon ECS 会自动将保留的标签 `AmazonECSCreated` 和 `AmazonECSManaged` 添加到附加的卷中。如果您从卷中移除这些标签，则 Amazon ECS 将无法代表您管理该卷。有关标记 Amazon EBS 卷的更多信息，请参阅标记 Amazon EBS 卷。有关标记 Amazon ECS 源的更多信息，请参阅标记 Amazon EC2 资源。
- 不支持通过包含分区的 Amazon EBS 卷的快照预调配卷。
- 附加到由服务管理的任务的卷不会被保留，并且总是会在任务终止时被删除。
- 您无法将 Amazon EBS 卷配置为附加到正在 AWS Outposts 上运行的 Amazon ECS 任务。

在 Amazon ECS 任务定义中将卷配置推迟到启动时间

要将 Amazon EBS 卷配置为附加到您的任务，您必须在任务定义中指定卷和挂载点配置和给卷命名。您还必须将 `configuredAtLaunch` 设置为 `true`，因为无法在任务定义中为附加配置 Amazon EBS 卷。反之，在部署期间可配置 Amazon EBS 卷进行附加。

以下任务定义在任务定义中显示 mountPoints 和 volumes 对象的语法。有关任务定义参数的更多信息，请参阅Amazon ECS 任务定义参数。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

要使用 AWS Command Line Interface（AWS CLI）注册任务定义，请将模板另存为 JSON 文件，然后将该文件作为 register-task-definition 命令的输入传递。

要使用 AWS Management Console 创建和注册任务定义，请参阅使用控制台创建 Amazon ECS 任务定义。

```
{
    "family": "mytaskdef",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/nginx/nginx:latest",
            "networkMode": "awsvpc",
           "portMappings": [
                {
                    "name": "nginx-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp",
                    "appProtocol": "http"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "/mount/ebs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
```

```
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
}
```

`mountPoints`

类型：对象数组

必需：否

容器中数据卷的挂载点。此参数将映射到 Docker Remote API 的创建容器部分中的 `Volumes` 以及 docker run 的 `--volume` 选项。

Windows 容器可在 `$env:ProgramData` 所在的驱动器上挂载整个目录。Windows 容器无法在其他驱动器上挂载目录，并且挂载点不能跨驱动器使用。您必须指定挂载点才能将 Amazon EBS 卷直接附加到 Amazon ECS 任务。

`sourceVolume`

类型：字符串

必需：是，当使用 `mountPoints` 时

要挂载的卷的名称。

`containerPath`

类型：字符串

必需：是，当使用 `mountPoints` 时

挂载卷的容器中的路径。

`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`。

`name`

类型：字符串

必需：否

卷的名称。最多允许 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` 对象的 `sourceVolume` 参数中引用。

`configuredAtLaunch`

类型：布尔值

必需：是，当您要使用将 EBS 卷直接附加到任务时。

指定卷在启动时是否可配置。如果设置为 `true`，您可以在运行独立任务或者创建或更新服务时配置卷。如果设置为 `true`，则无法在任务定义中提供其他卷配置。必须提供此参数并将其设置为 `true` 才能将 Amazon EBS 卷配置为附加到一个任务中。

加密存储在适用于 Amazon ECS 的 Amazon EBS 卷中的数据

您可以使用 AWS Key Management Service（AWS KMS）来创建和管理用于保护您的数据的加密密钥。Amazon EBS 卷通过使用 AWS KMS keys 进行静态加密。将加密以下类型的数据：

- 在卷上静态存储的数据
- 磁盘 I/O
- 从卷中创建的快照
- 从快照中创建的新卷

您可以默认配置 Amazon EBS 加密，以便使用您为账户配置的 KMS 密钥，对创建和附加到任务的所有新卷进行加密。有关 Amazon EBS 加密和默认加密的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EBS 加密。

附加到任务的 Amazon EBS 卷可以通过使用别名为 `alias/aws/ebs` 的默认 AWS 托管式密钥 或对称的客户托管密钥进行加密。默认的 AWS 托管式密钥 对每个 AWS 区域 的每个 AWS 账户 都是唯一的，并且将自动创建。要创建对称的客户托管密钥，请按照《AWS KMS 开发人员指南》的创建对称加密 KMS 密钥中的步骤进行操作。

客户托管的 KMS 密钥政策

要使用您的客户托管密钥对附加到您的任务的 EBS 卷进行加密，您必须配置 KMS 密钥政策，以确保用于卷配置的 IAM 角色拥有使用该密钥的必要权限。密钥政策必须包含 `kms:CreateGrant` 和

kms:GenerateDataKey* 权限。kms:ReEncryptTo 和 kms:ReEncryptFrom 权限是加密使用快照创建的卷所必需的。如果您只想为附加配置和加密新的空卷，您可以排除 kms:ReEncryptTo 和 kms:ReEncryptFrom 权限。

以下 JSON 代码段显示您可以附加到 KMS 密钥政策的密钥政策语句。使用这些语句将允许 ECS 使用密钥来加密 EBS 卷。要使用策略语句示例，请将 *user input placeholders* 替换为您自己的信息。与往常一样，只配置您需要的权限。

```
{
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:DescribeKey",
      "Resource":"*"
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": [
      "kms:GenerateDataKey*",
      "kms:ReEncryptTo",
      "kms:ReEncryptFrom"
      ],
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:CreateGrant",
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
```

```
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        },
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
```

有关密钥策略和权限的更多信息，请参阅《AWS KMS 开发人员指南》中的 AWS KMS 中的密钥策略和 AWS KMS 权限。要排查与密钥权限相关的 EBS 卷附加问题，请参阅排查 Amazon ECS 任务挂载 Amazon EBS 卷的问题 。

指定 Amazon ECS 部署时的 Amazon EBS 卷配置

将 `configuredAtLaunch` 参数设置为 `true` 注册任务定义后，您可以在运行独立任务或创建或更新服务时在部署时配置 Amazon EBS 卷。

要配置卷，您可以使用 Amazon ECS API，也可以将 JSON 文件作为以下 AWS CLI 命令的输入传递：

- 运行独立 ECS 任务的 `run-task`。
- 在特定的容器实例中运行独立 ECS 任务的 `start-task`。此命令不适用于 Fargate 启动类型任务。
- 创建新的 ECS 服务的 `create-service`。
- 更新现有服务的 `update-service`。

> **Note**
>
> 要使任务中的容器写入已挂载的 Amazon EBS 卷，您必须以根用户身份运行该容器。

您也可以使用 AWS Management Console 来配置 Amazon EBS 卷。有关更多信息，请参阅 将应用程序作为 Amazon ECS 任务运行、使用控制台创建 Amazon ECS 服务 和 使用控制台更新 Amazon ECS 服务。

以下 JSON 代码段显示可在部署时配置的 Amazon EBS 卷的所有参数。要使用这些参数进行卷配置，请将 *`user input placeholders`* 替换为您自己的信息。有关这些参数的更多信息，请参阅卷配置。

```
"volumeConfigurations": [
        {
            "name": "ebs-volume",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "volumeType": "gp3",
                "sizeInGiB": 10,
                "snapshotId": "snap-12345",
                "iops": 3000,
                "throughput": 125,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "arn:aws::iam:1111222333:role/ecsInfrastructureRole",
                 "terminationPolicy": {
                    "deleteOnTermination": true//can't be configured for service-
managed tasks, always true
                },
                "filesystemType": "ext4"
            }
        }
    ]
```

Important

确保您在配置中指定的 volumeName 与您在任务定义中指定的 volumeName 相同。

有关检查卷附加状态的信息，请参阅排查 Amazon ECS 任务挂载 Amazon EBS 卷的问题 。有关 EBS 卷附加所需的 Amazon ECS 基础设施 AWS Identity and Access Management（IAM）角色的信息，请参阅Amazon ECS 基础设施 IAM 角色。

以下是显示 Amazon EBS 卷配置的 JSON 代码段示例。这些示例可以通过将代码段保存在 JSON 文件中并将文件作为 AWS CLI 命令的参数（使用 `--cli-input-json file://filename` 参数）传递来使用。将 *user input placeholders* 替换为您自己的信息。

为独立任务配置卷

以下代码段显示配置 Amazon EBS 卷以附加到独立任务的语法。以下 JSON 代码段显示配置 `volumeType`、`sizeInGiB`、`encrypted` 和 `kmsKeyId` 设置的语法。JSON 文件中指定的配置用于创建 EBS 卷并将其附加到独立任务。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

在创建服务时配置卷

以下代码段显示配置 Amazon EBS 卷以附加到服务管理的任务的语法。这些卷是通过使用 `snapshotId` 从快照中获取的。JSON 文件中指定的配置用于创建 EBS 卷并将其附加到服务管理的每个任务。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
```

```
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
              "snapshotId": "snap-12345"
            }
        }
    ]
}
```

在更新服务时配置卷

以下 JSON 代码段显示更新以前未将 Amazon EBS 卷配置为附加到任务的服务的语法。您必须提供任务定义修订的 ARN，其 `configuredAtLaunch` 被设置为 `true`。以下 JSON 代码段显示配置 `volumeType`、`sizeInGiB`、`throughput` 和 `iops` 以及 `filesystemType` 设置的语法。此配置用于创建 EBS 卷并将其附加到服务管理的每个任务。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
  "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
               "volumeType": "gp3",
                "sizeInGiB": 100,
                 "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
    ]
}
```

将服务配置为不再利用 Amazon EBS 卷

以下 JSON 代码段显示更新服务以不再利用 Amazon EBS 卷的语法。您必须提供 `configuredAtLaunch` 被设置为 `false` 的任务定义的 ARN，或者不带 `configuredAtLaunch` 参数的任务定义。您还必须提供一个空的 `volumeConfigurations` 对象。

```
{
```

```
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "serviceName": "mysvc",
    "desiredCount": 2,
    "volumeConfigurations": []
}
```

Amazon EBS 卷的终止策略

当 Amazon ECS 任务终止时，Amazon ECS 使用 `deleteOnTermination` 值来确定是否应删除与已终止的任务关联的 Amazon EBS 卷。默认情况下，任务终止时，将会删除附加到任务的 EBS 卷。对于独立任务，您可以更改此设置，改为在任务终止时保留卷。

> **Note**
>
> 附加到由服务管理的任务的卷不会被保留，并且总是会在任务终止时被删除。

标记 Amazon EBS 卷

您可以使用 `tagSpecifications` 对象标记 Amazon EBS 卷。使用该对象，您可以提供自己的标签，并从任务定义或服务中设置标签的传播，具体取决于卷是附加到一项独立任务还是附加到服务中的某项任务。可以附加到一个卷上的最大标签数为 50。

> **Important**
>
> Amazon ECS 会自动将保留的 `AmazonECSCreated` 和 `AmazonECSManaged` 标签附加到 Amazon EBS 卷中。这意味着您可以控制最多将 48 个其他标签附加到卷的情况。这些附加标签可以是用户定义的、ECS 管理的或传播的标签。

如果您想向卷中添加 Amazon ECS 托管的标签，则必须在 `UpdateService`、`CreateService`、`RunTask` 或 `StartTask` 调用中将 `enableECSManagedTags` 设置为 `true`。如果您启用 Amazon ECS 管理的标签，Amazon ECS 将使用集群和服务信息（`aws:ecs:`*`clusterName`* 和 `aws:ecs:`*`serviceName`*）自动标记卷。有关标记 Amazon ECS 源的更多信息，请参阅标记 Amazon EC2 资源。

以下 JSON 代码段显示使用用户定义的标签标记附加到服务中每个任务的每个 Amazon EBS 卷的语法。要使用此示例创建服务，请将 *`user input placeholders`* 替换为您自己的信息。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "enableECSManagedTags": true,
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                 "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ]
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

> **Important**
>
> 您必须指定 volume 资源类型来标记 Amazon EBS 卷。

Fargate 按需任务的 Amazon EBS 卷的性能

可用于 Fargate 按需任务的基准 Amazon EBS 卷 IOPS 和吞吐量取决于您为任务请求的总 CPU 单位数。如果您为 Fargate 任务请求 0.25、0.5 或 1 个虚拟 CPU 单位（vCPU），建议您配置通用型 SSD

卷（gp2 或 gp3）或硬盘驱动器（HDD）卷（st1 或 sc1）。如果您为 Fargate 任务请求 1 个以上 vCPU，则以下基准性能限制适用于附加到该任务的 Amazon EBS 卷。您可能会暂时获得比以下限制更高的 EBS 性能。但是，建议您根据这些限制来计划工作负载。

请求的 CPU 单位数（以 vCPU 为单位）	基准 Amazon EBS IOPS（16 KiB I/O）	基准 Amazon EBS 吞吐量（以 MiBps 为单位，128 KiB I/O）	基准带宽（以 Mbps 为单位）
2	3000	75	360
4	5000	120	1,150
8	10000	250	2,300
16	15000	500	4,500

> **Note**
>
> 当您配置 Amazon EBS 卷以附加到 Fargate 任务时，Fargate 任务的 Amazon EBS 性能限制将在任务的临时存储和附加的卷之间共享。

排查 Amazon ECS 任务挂载 Amazon EBS 卷的问题

您可能需要对 Amazon EBS 卷与 Amazon ECS 任务的连接进行问题排查或验证。

检查卷附加状态

您可以使用 AWS Management Console 来查看 Amazon EBS 卷附加到 Amazon ECS 任务的状态。如果任务开始但附加失败，则您还会看到一个状态原因，可以用它来进行故障排除。创建的卷将被删除，且任务将停止。有关状态原因的更多信息，请参阅 Amazon EBS 卷附加到 Amazon ECS 任务的状态原因。

要使用控制台查看卷的附加状态和状态原因

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在集群页面上，选择您的任务在其中运行的集群。此时会显示集群的详细信息页面。
3. 在集群的详细信息页面上，选择任务选项卡。

4. 选择要查看其卷挂载状态的任务。如果要检查的任务已停止，则可能需要使用筛选所需状态并选择已停止。
5. 在任务的详细信息页面上，选择卷选项卡。您将能够在附加状态下看到 Amazon EBS 卷的附加状态。如果卷无法附加到任务，则可以在附加状态下选择状态以显示失败原因。

您还可以使用 DescribeTasks API 查看任务的卷附加状态和关联的状态原因。

服务和任务失败

您可能会遇到非特定于 Amazon EBS 卷的服务或任务故障，这些故障可能会影响卷附加。有关更多信息，请参阅

- 服务事件消息
- 已停止的任务错误代码
- API 失败原因

Amazon EBS 卷附加到 Amazon ECS 任务的状态原因

使用以下参考来修复您在配置 Amazon EBS 卷以附加到 Amazon ECS 任务时，在 AWS Management Console 中可能遇到的以状态原因形式出现的问题。有关在控制台中查找这些状态原因的更多信息，请参阅检查卷附加状态。

ECS 无法担任配置的 ECS 基础设施角色“arn:aws:iam::*111122223333*:role/*ecsInfrastructureRole*”。请确认传递的角色与 Amazon ECS 之间存在适当的信任关系

此状态原因显示在以下场景中。

- 您提供没有附加必要的信任策略的 IAM 角色。如果 Amazon ECS 基础设施 IAM 角色没有必要的信任策略，则 Amazon ECS 无法访问该角色。任务可能会卡在 DEPROVISIONING 状态。有关必要信任策略的更多信息，请参阅Amazon ECS 基础设施 IAM 角色。
- 您的 IAM 用户无权将 Amazon ECS 基础设施角色传递给 Amazon ECS。任务可能会卡在 DEPROVISIONING 状态。为避免出现此问题，您可以将 PassRole 权限附加到您的用户。有关更多信息，请参阅 Amazon ECS 基础设施 IAM 角色。
- 您的 IAM 角色没有附加 Amazon EBS 卷的必要权限。任务可能会卡在 DEPROVISIONING 状态。有关将 Amazon EBS 卷附加到任务所需的特定权限的更多信息，请参阅Amazon ECS 基础设施 IAM 角色。

> **Note**
>
> 您也可能会因角色传播延迟看到此错误消息。如果在等待几分钟后重试使用该角色仍无法解决问题，则可能错误地配置了角色的信任策略。

ECS 未能设置 EBS 卷。遇到 IdempotentParameterMismatch”；“您提供的客户端令牌与已删除的资源相关联。请使用其他客户端令牌。”

以下 AWS KMS 密钥场景可能会导致出现 `IdempotentParameterMismatch` 消息：

- 您指定的 KMS 密钥 ARN、ID 或别名无效。在这种情况下，任务可能看起来成功启动，但由于 AWS 对 KMS 密钥进行了异步身份验证，因此任务最终会失败。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EBS 加密。
- 您提供的客户托管密钥缺少允许 Amazon ECS 基础设施 IAM 角色使用该密钥进行加密的权限。为避免出现密钥政策权限问题，请参阅 Amazon EBS 卷的数据加密中的 AWS KMS 密钥政策示例。

您可以将 Amazon EventBridge 设置为向目标（例如 Amazon CloudWatch 组）发送 Amazon EBS 卷事件和 Amazon ECS 任务状态更改事件。然后，您可以使用这些事件来识别影响卷附加的特定客户托管密钥相关问题。有关更多信息，请参阅

- AWS re:Post 上的如何创建 CloudWatch 日志组以用作 EventBridge 规则的目标？。
- 任务状态更改事件。
- 《Amazon EBS 用户指南》中的 Amazon EBS 的 EventBridge。

在配置 EBS 卷到任务的附加时，ECS 超时。

以下文件系统格式场景将产生此消息。

- 您在配置期间指定的文件系统格式与任务的操作系统不兼容。
- 您将 Amazon EBS 卷配置为从快照中创建，但快照的文件系统格式与任务的操作系统不兼容。对于从快照创建的卷，必须指定创建快照时卷使用的相同文件系统类型。

您可以利用 Amazon ECS 容器代理日志对 Amazon EC2 启动类型任务的此消息进行问题排查。有关更多信息，请参阅 Amazon ECS 日志文件位置和 Amazon ECS 日志收集器。

将 Amazon EFS 卷与 Amazon ECS 结合使用

Amazon Elastic File System (Amazon EFS) 提供简单的可扩展文件存储以供您的 Amazon ECS 任务使用。使用 Amazon EFS 时，存储容量是弹性的。它会随着您添加和删除文件而自动增加和缩减。您的应用程序可在需要时获得所需存储。

您可以将 Amazon EFS 文件系统与 Amazon ECS 配合使用，以便导出跨容器实例队列的文件系统数据。这样，无论您的任务登录的是哪个实例，都可以访问相同的持久性存储。您的任务定义必须引用容器实例上的卷挂载才能使用该文件系统。

有关教程，请参阅 使用控制台为 Amazon ECS 配置 Amazon EFS 文件系统。

注意事项

使用 Amazon EFS 卷时应考虑以下事项：

- 对于使用 EC2 启动类型的任务，已将 Amazon EFS 文件系统支持作为一个公共预览版添加，其中包括经 Amazon ECS 优化的 AMI 版本 20191212 以及容器代理版本 1.35.0。但是，Amazon EFS 文件系统支持通过经 Amazon ECS 优化的 AMI 版本 20200319 和容器代理版本1.38.0 正式推出，该版本包含 Amazon EFS 访问点和 IAM 授权功能。我们建议您使用经 Amazon ECS 优化的 AMI 版本 20200319 或更高版本以利用这些功能。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。

  > **Note**
  >
  > 如果您创建自己的 AMI，则必须使用容器代理 1.38.0 或更高版本、ecs-init 1.38.0-1 或更高版本，并在 Amazon EC2 实例上运行以下命令以启用 Amazon ECS 卷插件。命令取决于您将 Amazon Linux 2 还是 Amazon Linux 用作基本映像。
  >
  > Amazon Linux 2
  >
  > ```
  > yum install amazon-efs-utils
  > systemctl enable --now amazon-ecs-volume-plugin
  > ```
  >
  > Amazon Linux
  >
  > ```
  > yum install amazon-efs-utils
  > sudo shutdown -r now
  > ```

- 对于 Fargate 托管的任务，平台版本 1.4.0 或更高版本（Linux）支持 Amazon EFS 文件系统。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。

- 在 Fargate 上托管的任务中使用 Amazon EFS 卷时，Fargate 将创建负责管理 Amazon EFS 卷的主管容器。主管容器使用少量的任务内存。主管容器在查询任务元数据版本 4 端点时可见。此外，它作为容器名称 `aws-fargate-supervisor` 在 CloudWatch Container Insights 中可见。有关使用 Amazon EC2 启动类型的更多信息，请参阅Amazon ECS 任务元数据端点版本 4。有关使用 Fargate 启动类型的更多信息，请参阅Fargate 上任务的 Amazon ECS 任务元数据端点版本 4。
- 不支持在外部实例上使用 Amazon EFS 卷或指定 `EFSVolumeConfiguration`。
- 建议您将代理配置文件中的 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 参数值设置为小于默认值（约 1 小时）。此更改有助于防止 EFS 挂载凭证过期，并允许清理未使用的挂载。有关更多信息，请参阅 Amazon ECS 容器代理配置。

使用 Amazon EFS 接入点

Amazon EFS 访问点是 EFS 文件系统中特定于应用程序的入口点，用于管理应用程序对共享数据集的访问。有关 Amazon EFS 访问点以及如何控制访问的更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 访问点。

接入点可以为通过接入点发出的所有文件系统请求强制执行用户身份（包括用户的 POSIX 组）。访问点还可以为文件系统强制执行不同的根目录。这样，客户端只能访问指定目录或其子目录中的数据。

> **Note**
>
> 创建 EFS 访问点时，请在文件系统上指定用作根目录的路径。在 Amazon ECS 任务定义中引用具有访问点 ID 的 EFS 文件系统时，必须忽略根目录或将根目录设置为 `/`，以便在 EFS 访问点上强制执行设置的路径。

您可以使用 Amazon ECS 任务 IAM 角色强制特定应用程序使用特定的访问点。通过将 IAM policy 与访问点相结合，您可以为您的应用程序提供对特定数据集的安全访问。有关如何使用任务 IAM 角色的更多信息，请参阅Amazon ECS 任务 IAM 角色。

将 Amazon EFS 卷与 Amazon ECS 结合使用的最佳实践

将 Amazon EFS 与 Amazon ECS 结合使用时，请记下以下最佳实践建议。

Amazon EFS 卷的安全性和访问控制

Amazon EFS 提供访问控制功能，您可以使用这些功能来确保存储在 Amazon EFS 文件系统中的数据是安全的，并且只能从需要它的应用程序中访问。您可以通过启用静态加密和动态加密来保护数据。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 Amazon EFS 中的数据加密。

除了数据加密之外，您还可以使用 Amazon EFS 来限制对文件系统的访问。可以通过三种方法在 EFS 中实现访问控制。

- 安全组 – 使用 Amazon EFS 挂载目标，您可以配置用于允许和拒绝网络流量的安全组。您可以将附加到 Amazon EFS 的安全组配置为允许来自附加到您的 Amazon ECS 实例的安全组的 NFS 流量（端口 2049），或者在使用 `awsvpc` 网络模式时，允许来自 Amazon ECS 任务的该流量。
- IAM – 您可以使用 IAM 限制对 Amazon EFS 文件系统的访问。配置后，Amazon ECS 任务需要一个 IAM 角色来访问文件系统，以挂载 EFS 文件系统。有关详细信息，请参阅 Amazon Elastic 文件系统用户指南 中的使用 IAM 控制文件系统数据访问。

  IAM 策略还可以强制执行预定义的条件，例如要求客户端使用 TLS 连接到 Amazon EFS 文件系统。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的客户端的 Amazon EFS 条件键。
- Amazon EFS 接入点 – Amazon EFS 接入点是 Amazon EFS 文件系统中特定于应用程序的入口点。您可以使用接入点为通过接入点发出的所有文件系统请求强制执行用户身份（包括用户的 POSIX 组）。访问点还可以为文件系统强制执行不同的根目录。这样，客户端只能访问指定目录或其子目录中的数据。

考虑在 Amazon EFS 文件系统上实施全部三种访问控制，以最大限度地提高安全性。例如，您可以将附加到 Amazon EFS 挂载点的安全组配置为仅允许来自与您的容器实例或 Amazon ECS 任务关联的安全组的入口 NFS 流量。此外，您可以将 Amazon EFS 配置为要求 IAM 角色访问文件系统，即使连接来自允许的安全组。最后，您可以使用 Amazon EFS 接入点，强制执行 POSIX 用户权限并为应用程序指定根目录。

以下任务定义代码段显示如何使用接入点挂载 Amazon EFS 文件系统。

```
"volumes": [
    {
      "efsVolumeConfiguration": {
        "fileSystemId": "fs-1234",
        "authorizationConfig": {
          "acessPointId": "fsap-1234",
          "iam": "ENABLED"
        },
        "transitEncryption": "ENABLED",
        "rootDirectory": ""
      },
      "name": "my-filesystem"
    }
```

```
]
```

Amazon EFS 卷的性能

Amazon EFS 提供两种性能模式：通用型模式和最大 I/O 模式。通用型模式适用于对延迟敏感的应用程序，例如内容管理系统和 CI/CD 工具。相比之下，最大 I/O 文件系统适用于数据分析、媒体处理和机器学习等工作负载。这些工作负载需要从数百甚至数千个容器中执行并行操作，并且需要尽可能高的总吞吐量和 IOPS。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 Amazon EFS 性能模式。

某些对延迟敏感的工作负载同时需要由最大 I/O 性能模式提供更高的 I/O 级别以及由通用性能模式提供的更低延迟。对于此类工作负载，我们建议创建多个通用性能模式文件系统。用此方式，您可以将应用程序工作负载分散到所有这些文件系统间，只要工作负载和应用程序可以支持它即可。

Amazon EFS 卷的吞吐量

所有 Amazon EFS 文件系统都有相关的计量吞吐量，该吞吐量由使用预调配吞吐量的文件系统的预调配吞吐量或使用突增吞吐量的文件系统以 EFS 标准或单区存储类存储的数据量决定。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的了解计量吞吐量。

Amazon EFS 文件系统的默认吞吐量模式为突增模式。在突增模式下，可用于文件系统的吞吐量会随着文件系统的增长而横向缩减或扩展。因为基于文件的工作负载通常会激增，在一段时间内需要高水平的吞吐量，其余时间则需要低水平的吞吐量，因此，Amazon EFS 被设计为可在一段时间内突增到允许高吞吐量。此外，由于许多工作负载的读取量大，因此读取操作与其他 NFS 操作（例如写入）的计量比例为 1:3。

所有 Amazon EFS 文件系统均为每 TB 的 Amazon EFS 标准存储或 Amazon EFS 单区存储提供 50 MB/s 的一致基准性能。所有文件系统，不管其大小如何，都能突增到 100 MB/s。拥有超过 1TB 的 EFS 标准存储或 EFS 单区存储的文件系统每 TB 可以突增至 100 MB/s。由于读取操作以 1:3 的比例计量，对于每 TiB 的读取吞吐量，您最多可以驱动到 300 MiB/s。当您将数据添加到文件系统时，文件系统可用的最大吞吐量会随着您在 Amazon EFS 标准存储类中的存储量而自动线性扩展。如果您需要的吞吐量超过存储的数据量所能达到的吞吐量，则可以将预配置吞吐量配置为工作负载需要的特定量。

文件系统吞吐量在连接到文件系统的所有 Amazon EC2 实例间共享。例如，一个吞吐量可突增至 100 MB/s 的 1TB 文件系统可以从单个 Amazon EC2 实例中驱动 100 MB/s，每个实例可以驱动 10 MB/s。有关更多信息，请参阅《Amazon Elastic 文件系统用户指南》中的 Amazon EFS 性能。

优化 Amazon EFS 卷的成本

Amazon EFS 为您简化扩展存储的过程。随着您添加更多数据，Amazon EFS 文件系统会自动增长。尤其是在 Amazon EFS 突增吞吐量模式下，Amazon EFS 上的吞吐量将随着标准存储类中文件系统

大小的增大而扩展。要在不为 EFS 文件系统的预调配吞吐量支付额外费用的情况下提高吞吐量，您可以与多个应用程序共享一个 Amazon EFS 文件系统。使用 Amazon EFS 接入点，您可以在共享的 Amazon EFS 文件系统中实施存储隔离。这样，即使应用程序仍共享同一个文件系统，但除非您授权，否则它们无法访问数据。

随着数据的增长，Amazon EFS 可帮助您将不经常访问的文件自动移动到较低的存储类。Amazon EFS 标准-不频繁访问（IA）存储类可降低并非每天访问的文件的存储成本。这样做并不会损害 Amazon EFS 提供的高可用性、高持久性、弹性和 POSIX 文件系统访问。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 Amazon EFS 存储类。

考虑使用 Amazon EFS 生命周期策略，通过将不经常访问的文件移至 Amazon EFS IA 存储来自动节省成本。有关更多信息，请参阅 Amazon Elastic File System User Guide（《Amazon Elastic File System 用户指南》）中的 Amazon EFS lifecycle management（Amazon EFS 生命周期管理）。

创建 Amazon EFS 文件系统时，您可以在 Amazon EFS 跨多个可用区（标准）复制您的数据或将您的数据冗余存储在单个可用区中之间进行选择。与 Amazon EFS 标准存储类相比，Amazon EFS 单区存储类可以大幅降低存储成本。对于不需要多可用区弹性的工作负载，考虑使用 Amazon EFS 单区存储类。您可以通过将不经常访问的文件移至 Amazon EFS 单区-不频繁访问来进一步降低 Amazon EFS 单区存储的成本。有关更多信息，请参阅 Amazon EFS 不频繁访问。

Amazon EFS 卷数据保护

对于使用标准存储类的文件系统，Amazon EFS 跨多个可用区冗余存储您的数据。如果您选择 Amazon EFS 单区存储类，则您的数据将冗余存储在单个可用区内。此外，Amazon EFS 被设计为在指定年度内提供 99.999999999%（11 个 9）的持久性。

与任何环境一样，最佳做法是进行备份并构建防范意外删除的保护措施。对于 Amazon EFS 数据，最佳实践包括使用 AWS Backup 进行正常运行且定期测试的备份。使用 Amazon EFS 单区存储类的文件系统默认配置为在创建文件系统时自动备份文件，除非您选择禁用此功能。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 Amazon EFS 的数据保护。

在 Amazon ECS 任务定义中指定 Amazon EFS 文件系统

要为容器使用 Amazon EFS 文件系统卷，您必须在任务定义中指定卷和挂载点配置。以下任务定义 JSON 代码段显示容器的 `volumes` 和 `mountPoints` 对象的语法。

```
{
    "containerDefinitions": [
        {
            "name": "container-using-efs",
            "image": "amazonlinux:2",
```

```
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEfsVolume",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1234",
                "rootDirectory": "/path/to/my/data",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": integer,
                "authorizationConfig": {
                    "accessPointId": "fsap-1234",
                    "iam": "ENABLED"
                }
            }
        }
    ]
}
```

`efsVolumeConfiguration`

类型：对象

必需：否

使用 Amazon EFS 卷时将指定此参数。

`fileSystemId`

类型：字符串

必需：是

要使用的 Amazon EFS 文件系统 ID。

`rootDirectory`

类型：字符串

必需：否

Amazon EFS 文件系统中要作为主机内的根目录挂载的目录。如果忽略此参数，将使用 Amazon EFS 卷的根目录。指定/与忽略此参数效果相同。

> **Important**
>
> 如果在 `authorizationConfig` 中指定了 EFS 访问点，则必须省略根目录值，或者将其设置为 `/`，以便在 EFS 访问点上强制执行设置的路径。

`transitEncryption`

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

指定是否对 Amazon ECS 主机和 Amazon EFS 服务器之间传输的 Amazon EFS 数据启用加密。如果使用 Amazon EFS IAM 授权，则必须启用传输加密。如果忽略此参数，将使用默认值`DISABLED`。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的加密传输中的数据。

`transitEncryptionPort`

类型：整数

必需：否

在 Amazon ECS 主机和 Amazon EFS 服务器之间发送加密数据时要使用的端口。如果未指定传输加密端口，将使用 Amazon EFS 挂载帮助程序使用的端口选择策略。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 EFS 挂载帮助程序。

`authorizationConfig`

类型：对象

必需：否

Amazon EFS 文件系统的授权配置详细信息。

`accessPointId`

类型：字符串

必需：否

要使用的接入点 ID。如果指定了访问点，则必须省略在 `efsVolumeConfiguration` 中指定的根目录值，或者将其设置为 `/`，以便在 EFS 访问点上强制执行设置的路径。如果使用接入点，则必须在`EFSVolumeConfiguration`中启用传输加密。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 接入点。

`iam`

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

指定挂载 Amazon EFS 文件系统时是否使用在任务定义中定义的 Amazon ECS 任务 IAM 角色。如果启用，则必须在`EFSVolumeConfiguration`中启用传输加密。如果忽略此参数，将使用默认值`DISABLED`。有关更多信息，请参阅适用于任务的 IAM 角色。

使用控制台为 Amazon ECS 配置 Amazon EFS 文件系统

了解如何将在 Amazon Elastic File System（Amazon EFS）文件系统与 Amazon ECS 结合使用。

步骤 1：创建 Amazon ECS 集群

请按照以下步骤创建 Amazon ECS 集群。

要创建新集群（Amazon ECS 控制台）

在开始之前，分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。

3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在集群配置下方的集群名称中，输入 `EFS-tutorial` 作为集群名称。
6. （可选）要更改任务和服务启动所在的 VPC 和子网，在 Networking（联网）下，执行以下任一操作：

   - 要删除子网，请在 Subnets（子网）下，为您要删除的每个子网选择 X。
   - 要更改为默认 VPC 以外的 VPC，在 VPC 下，选择现有的 VPC，然后在 Subnets（子网）中，选择每个子网。
7. 要向集群添加 Amazon EC2 实例，请展开基础设施，然后选择 Amazon EC2 实例。接下来，配置充当容量提供程序的 自动扩缩组：

   - 要创建 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择 Create new group（创建新组），然后提供有关组的以下详细信息：

     - 对于操作系统/架构，选择 Amazon Linux 2。
     - 对于 EC2 instance type (EC2 实例类型)，选择 `t2.micro`。

       对于 SSH key pair（SSH 密钥对），连接到实例时，选择可证明您身份的密钥对。
     - 对于容量，输入 1。
8. 选择创建。

步骤 2：为 Amazon EC2 实例和 Amazon EFS 文件系统创建安全组

在此步骤中，您需要为 Amazon EFS 文件系统创建一个安全组，以允许端口 80 上的入站网络流量，并为 Amazon EFS 文件系统创建一个安全组，以允许您的容器实例进行入站访问。

使用以下选项为您的 Amazon EC2 实例创建安全组：

- 安全组名称 - 安全组的唯一名称。
- VPC - 您之前为集群确定的 VPC。
- 入站规则
  - 类型 - HTTP
  - 来源 - 0.0.0.0/0。

使用以下选项为您的 Amazon EFS 文件系统创建安全组：

- 安全组名称 - 安全组的唯一名称。例如，`EFS-access-for-sg-dc025fa2`。
- VPC - 您之前为集群确定的 VPC。
- 入站规则
  - 类型 - NFS
  - 来源 - 使用您为实例创建的安全组的 ID 自定义。

有关如何创建安全组的信息，请参阅《Amazon EC2 用户指南》中的创建安全组。

步骤 3：创建 Amazon EFS 文件系统

在该步骤中，您将创建一个 Amazon EFS 文件系统。

为 Amazon ECS 任务创建 Amazon EFS 文件系统。

1. 访问 https://console.aws.amazon.com/efs/，打开 Amazon Elastic File System 控制台。
2. 选择 Create file system。
3. 输入文件系统的名称，然后选择托管容器实例的 VPC。默认情况下，指定 VPC 中的每个子网都会收到使用该 VPC 默认安全组的挂载目标。然后，选择自定义。

   > **Note**
   >
   > 此教程假设您的 Amazon EFS 文件系统、Amazon ECS 集群、容器实例和任务位于同一 VPC 中。有关从其他 VPC 挂载文件系统的更多信息，请参阅《Amazon EFS 用户指南》中的演练：从其他 VPC 挂载文件系统。

4. 在文件系统设置页面上，配置可选设置，然后在性能设置下，为您的文件系统选择突增吞吐量模式。配置完设置后，选择下一步。

   a. (可选) 为文件系统添加标签。例如，通过在 Name 键旁边的 Value 列中输入名称可以为文件系统指定唯一的名称。

   b. （可选）启用生命周期管理可节省不经常访问的存储的成本。有关更多信息，请参阅 Amazon Elastic File System 用户指南中的 EFS 生命周期管理。

   c. （可选）启用加密。选中该复选框可对 Amazon EFS 文件系统启用静态加密。

5. 在网络访问页面的挂载目标下，将每个可用区的现有安全组配置替换为您在 步骤 2：为 Amazon EC2 实例和 Amazon EFS 文件系统创建安全组 中为文件系统创建的安全组，然后选择下一步。
6. 您无需为本教程配置文件系统策略，因此您可以通过选择下一步来跳过本部分。

7. 查看文件系统选项，然后选择创建以完成此过程。
8. 从文件系统屏幕中，记录文件系统 ID。在下一步中，您将在 Amazon ECS 任务定义中引用此值。

步骤 4：向 Amazon EFS 文件系统添加内容

在此步骤中，您将 Amazon EFS 文件系统挂载到 Amazon EC2 实例并向其添加内容。这在本教程中用于测试，目的是说明数据的持久性质。使用此功能时，您通常会使用您的应用程序或其他方法将数据写入 Amazon EFS 文件系统。

创建 Amazon EC2 实例并挂载 Amazon EFS 文件系统

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择 Launch Instance（启动实例）。
3. 在应用程序和操作系统映像（Amazon 系统映像）下，选择 Amazon Linux 2 AMI（HVM）。
4. 在实例类型下，保留默认实例类型 `t2.micro`。
5. 在密钥对（登录）下，选择用于 SSH 访问实例的密钥对。
6. 在网络设置下，选择您为 Amazon EFS 文件系统和 Amazon ECS 集群指定的 VPC。选择子网和在 步骤 2：为 Amazon EC2 实例和 Amazon EFS 文件系统创建安全组 中创建的实例安全组。配置实例的安全组。确保已启用自动分配公有 IP。
7. 在配置存储下，选择文件系统的编辑按钮，然后选择 EFS。选择您在 步骤 3：创建 Amazon EFS 文件系统 中创建的文件系统。您可以选择更改挂载点或保留默认值。

   > **Important**
   >
   > 您必须先选择子网，然后才能将文件系统添加到实例中。

8. 清除自动创建和附加安全组。将另一个复选框保持选中状态。请选择 Add shared file system（添加共享文件系统）。
9. 在 Advanced Details (高级详细信息) 下，确保通过 Amazon EFS 文件系统挂载步骤自动填充用户数据脚本。
10. 在摘要下，确保实例数为 1。选择启动实例。
11. 在启动状态页面上，选择查看所有实例以查看实例的状态。最初，实例状态为 `PENDING`。在状态变为 `RUNNING` 且实例通过所有状态检查后，实例就可以使用了。

现在，您可以连接到 Amazon EC2 实例并向 Amazon EFS 文件系统添加内容。

连接到 Amazon EC2 实例并向 Amazon EFS 文件系统添加内容

1. SSH 到您创建的 Amazon EC2 实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的连接到 Linux 实例。
2. 从终端窗口，运行 df -T 命令以验证 Amazon EFS 文件系统是否已挂载。在下面的输出中，我们突出了 Amazon EFS 文件系统挂载。

```
$ df -T
Filesystem     Type            1K-blocks    Used        Available Use% Mounted on
devtmpfs       devtmpfs           485468       0           485468   0% /dev
tmpfs          tmpfs              503480       0           503480   0% /dev/shm
tmpfs          tmpfs              503480     424           503056   1% /run
tmpfs          tmpfs              503480       0           503480   0% /sys/fs/
cgroup
/dev/xvda1     xfs               8376300 1310952          7065348  16% /
127.0.0.1:/    nfs4     9007199254739968       0 9007199254739968   0% /mnt/efs/fs1
tmpfs          tmpfs              100700       0           100700   0% /run/
user/1000
```

3. 导航到挂载 Amazon EFS 文件系统的目录。在上述示例中，即为 /mnt/efs/fs1。
4. 使用以下内容创建名为 index.html 的文件：

```
<html>
    <body>
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container
 storage.</p>
    </body>
</html>
```

步骤 5：创建任务定义

以下任务定义创建名为 efs-html 的数据卷。nginx 容器将该主机数据卷挂载在 NGINX 根目录 (/usr/share/nginx/html) 下。

要使用 Amazon ECS 控制台创建新的任务定义

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。

3. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
4. 在 JSON 编辑器框中，复制并粘贴以下 JSON 文本，将 `fileSystemId` 替换为 Amazon EFS 文件系统的 ID。

```
{
    "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": [
        {
            "name": "efs-html",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1324abcd",
                "transitEncryption": "ENABLED"
            }
        }
    ],
    "family": "efs-tutorial",
    "executionRoleArn":"arn:aws::iam::111122223333:role/ecsTaskExecutionRole"
}
```

> **Note**
>
> 您可以将以下权限添加到您的 Amazon ECS 任务执行 IAM 角色中，以允许 Amazon ECS 代理在启动时找到，并将 Amazon EFS 文件系统挂载到任务中。
>
> - `elasticfilesystem:ClientMount`
> - `elasticfilesystem:ClientWrite`
> - `elasticfilesystem:DescribeMountTargets`
> - `elasticfilesystem:DescribeFileSystems`

5. 选择创建。

步骤 6：运行任务并查看结果

现在，您的 Amazon EFS 文件系统已经创建，并且已经有 NGINX 容器要提供的 Web 内容，您可以使用您创建的任务定义运行任务了。NGINX Web 服务器提供简单的 HTML 页面。如果更新 Amazon EFS 文件系统中的内容，更改会传播到挂载了该文件系统的所有容器。

该任务在您为集群定义的子网中运行。

使用控制台运行任务并查看结果

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在 Clusters（集群）页面上，选择包含要运行独立任务的集群。

   确定要从其中启动服务的资源。

从中启动服务	步骤
集群	a. 在集群页面上，选择要在其中创建服务的集群。 b. 从任务选项卡上，选择运行新任务。
启动类型	a. 在 Task（任务）页面上，选择任务定义。

从中启动服务	步骤
	b. 如果有多个修订版，请选择相应的修订版。 c. 选择 Create（创建）、Run task（运行任务）。

3. （可选）选择计划任务在集群基础设施中的分发方式。展开 Compute configuration（计算配置），然后执行以下操作：

分配方式	步骤
启动类型	a. 在 Compute options（计算选项）部分中，选择 Launch type（启动类型）。 b. 对于启动类型，选择 EC2。

4. 对于应用程序类型，选择任务。
5. 对于任务定义，选择您之前创建的 `efs-tutorial` 任务定义。
6. 对于所需任务，输入 1。
7. 选择创建。
8. 在集群页面上，选择基础设施。
9. 在容器实例下，选择要连接的容器实例。
10. 在容器实例页面的联网下，记录您的实例的公有 IP。
11. 打开浏览器并输入公有 IP 地址。您将看到以下消息：

```
It works!
You are using an Amazon EFS file system for persistent container storage.
```

> **Note**
>
> 如果没有看到该消息，请确保容器实例的安全组允许端口 80 上的入站网络流量，且文件系统的安全组允许从容器实例进行入站访问。

将 FSx for Windows File Server 卷与 Amazon ECS 结合使用

FSx for Windows File Server 提供完全托管的 Windows 文件服务器，由 Windows 文件系统提供支持。使用 FSx for Windows File Server 和 ECS 时，您可以使用永久、分布式、共享的静态文件存储来配置 Windows 任务。有关更多信息，请参阅适用于 Windows File Server 的 FSx 定义。

> **Note**
>
> 使用经 Amazon ECS 优化的 Windows Server 2016 的完整 AMI 的 EC2 实例不支持 FSx for Windows File Server ECS 任务卷。
> 在 Fargate 配置中，您不能在 Windows 容器中使用 FSx for Windows File Server 卷。相反，您可以修改容器以在启动时挂载它们。

您可以使用 FSx for Windows File Server 来部署需要访问共享外部存储、高可用区域存储或高吞吐量存储的 Windows 工作负载。您可以将一个或多个 FSx for Windows File Server 文件系统卷挂载到 Amazon ECS Windows 实例上运行的 Amazon ECS 容器中。在单个 Amazon ECS 任务中，您可以在多个 Amazon ECS 容器之间共享 FSx for Windows File Server 文件系统卷的 FSX。

要使用 FSx for Windows File Server 和 ECS，在任务定义中包含 FFSx for Windows File Server 文件系统 ID 和相关信息。它们在以下示例任务定义 JSON 代码段中。在创建和运行任务定义之前，您需要执行以下操作。

- 已加入到有效域的 ECS Windows EC2 实例。它可以由 AWS Directory Service for Microsoft Active Directory、本地 Active Directory 或 Amazon EC2 上的自行托管的 Active Directory 托管。
- AWS Secrets Manager 密钥或 Systems Manager 参数，其中包含用于加入 Active Directory 域和附加 FSx for Windows File Server 文件系统的凭证。凭据值是您在创建 Active Directory 时输入的名称和密码凭据。

有关相关教程，请参阅 了解为 Amazon ECS 配置适用于 Windows File Server 的 FSx 文件系统。

注意事项

使用 FSx for Windows File Server 卷时，请考虑以下事项：

- 使用 Amazon ECS 的 FSx for Windows File Server 仅支持 Windows Amazon EC2 实例。不支持 Linux Amazon EC2 实例。
- 使用 Amazon ECS 的 FSx for Windows File Server 不支持 AWS Fargate。
- 使用 Amazon ECS 的 FSx for Windows File Server 使用 `awsvpc` 网络模式需要版本 `1.54.0` 或更高版本的容器代理。
- 可用于 Amazon ECS 任务的最大驱动器号数为 23。每个具有 FSx for Windows File Server 卷都将获得分配给它的驱动器号。
- 默认情况下，任务资源清理时间为任务结束后的三个小时。即使没有任务使用它，由任务创建的文件映射仍将持续 3 小时。可以通过使用 Amazon ECS 环境变量来配置`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 原定设置清理时间。有关更多信息，请参阅 Amazon ECS 容器代理配置。
- 任务通常仅在 FSx for Windows File Server 文件系统相同的 VPC 中运行。但是，如果 Amazon ECS 集群 VPC 与 FSx for Windows File Server 文件系统之间通过 VPC 对等连接建立了网络连接，则可以实现跨 VPC 的支持。
- 通过配置 VPC 安全组，可以在网络级别控制对 FSx for Windows File Server 文件系统的访问。只有通过正确配置的 Active Directory 安全组加入 Active Directory 域的 EC2 实例上托管的任务才能访问 FSx for Windows File Server 文件共享。如果安全组配置错误，Amazon ECS 将导致任务启动失败，并显示以下错误消息：`unable to mount file system` *`fs-id`*。”
- FSx for Windows File Server 集成 AWS Identity and Access Management (IAM) 来控制您的 IAM 用户和组可以对 FSx for Windows File Server 资源执行的操作。通过客户端授权，客户可以定义 IAM 角色，这些角色允许或拒绝对 FSx for Windows File Server 文件系统的访问，可以选择要求只读访问，也可以选择允许或不允许客户端对文件系统的根访问。有关更多信息，请参阅 Amazon FSx Windows 用户指南中的安全性。

将 FSx for Windows File Server 与 Amazon ECS 结合使用的最佳实践

将 FSx for Windows File Server 与 Amazon ECS 结合使用时，请记下以下最佳实践建议。

FSx for Windows File Server 的安全和访问控制

FSx for Windows File Server 提供以下访问控制功能，您可以使用这些功能来确保存储在 FSx for Windows File Server 文件系统中的数据是安全的，并且只能从需要它的应用程序中访问。

FSx for Windows File Server 卷的数据加密

FSx for Windows File Server 支持两种形式的文件系统加密。它们是传输中数据的加密和静态加密。在支持 SMB 协议 3.0 或更高版本的容器实例上映射的文件共享支持传输中数据加密。创建 Amazon FSx 文件系统时，系统会自动启用静态数据加密。Amazon FSx 会在您访问文件系统时使用 SMB 加密自动加密传输中数据，而无需修改应用程序。有关更多信息，请参阅《适用于 Windows File Server 的 Amazon FSx 用户指南》中的 Amazon FSx 中的数据加密。

使用 Windows ACL 进行文件夹级别的访问控制

Windows Amazon EC2 实例使用 Active Directory 凭证访问 Amazon FSx 文件共享。它使用标准的 Windows 访问控制列表（ACL）进行精细的文件和文件夹级别的访问控制。您可以创建多个凭证，每个凭证用于共享中映射到特定任务的特定文件夹。

在以下示例中，任务可以使用保存在 Secrets Manager 中的凭证访问文件夹 `App01`。它的 Amazon 资源名称（ARN）为 `1234`。

```
"rootDirectory": "\\path\\to\\my\\data\App01",
"credentialsParameter": "arn-1234",
"domain": "corp.fullyqualified.com",
```

在另一个示例中，任务可以使用保存在 Secrets Manager 中的凭证访问文件夹 `App02`。其 ARN 为 `6789`。

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
"domain": "corp.fullyqualified.com",
```

在 Amazon ECS 任务定义中指定适用于 Windows File Server 的 FSx 文件系统

要为容器使用 FSx for Windows File Server 文件系统卷，请在任务定义中指定卷和挂载点配置。以下任务定义 JSON 代码段显示容器的 `volumes` 和 `mountPoints` 对象的语法。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
```

```
            "command": ["New-Item -Path C:\\fsx-windows-dir\\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>It Works!</h2> <p>You are using Amazon
 FSx for Windows File Server file system for persistent container storage.</p>' -
Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force; Start-
Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -Destination C:\
\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
```

```
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

`FSxWindowsFileServerVolumeConfiguration`

类型：对象

必需：否

当您使用 FSx for Windows File Server 文件系统进行任务存储时，指定此参数。

`fileSystemId`

类型：字符串

必需：是

要使用的 FSx for Windows File Server 文件系统 ID。

`rootDirectory`

类型：字符串

必需：是

FSx for Windows File Server 文件系统中要作为主机内的根目录挂载的目录。

`authorizationConfig`

`credentialsParameter`

类型：字符串

必需：是

授权凭据选项：

- Secrets Manager 密钥的 Amazon Resource Name (ARN)。
- Secrets Manager 参数的 Amazon Resource Name (ARN)。

`domain`

类型：字符串

必需：是

由 AWS Directory Service for Microsoft Active Directory（AWS Managed Microsoft AD）目录托管的完全限定域名或自托管 EC2 Active Directory。

存储 FSx for Windows File Server 卷凭证的方法

存储凭据的凭据有两种不同的方法，以便与凭据参数一起使用。

- AWS Secrets Manager 密钥

  此凭据可以在 AWS Secrets Manager 控制台中使用其他密钥类型类别。您可以为每个键/值对、用户名/管理员和密码/*password* 添加一行。
- Systems Manager 参数

  通过以下示例代码片段中的形式输入文本，可以在 Systems Manager 参数控制台中创建此凭证。

```
{
  "username": "admin",
  "password": "password"
}
```

任务定义 `FSxWindowsFileServerVolumeConfiguration` 参数中的 `credentialsParameter` 将保存密钥 ARN 或 Systems Manager 参数 ARN。有关更多信息，请参阅 Secrets Manager用户指

南中的什么是 AWS Secrets Manager以及 Systems Manager 用户指南中的 Systems Manager 参数存储。

了解为 Amazon ECS 配置适用于 Windows File Server 的 FSx 文件系统

了解如何启动经 Amazon ECS 优化的 Windows 实例，该实例托管 FSx for Windows File Server 文件系统和可访问该文件系统的容器。要执行此操作，您首先创建一个AWS Directory Service AWS托管 Microsoft Active Directory。然后，您可以创建 FSx for Windows File Server 文件系统和带有 Amazon EC2 实例和任务定义的集群。您可以配置容器的任务定义使用 FSx for Windows File Server 文件系统。最后，您测试文件系统。

每次启动或删除 Active Directory 或 FSx for Windows File Server 文件系统时都需要 20 到 45 分钟。准备至少保留 90 分钟以完成本教程或通过几个课程完成本教程。

教程的先决条件

- 管理用户。请参阅 设置以使用 Amazon ECS。
- （可选）一个 PEM 密钥对，用于通过 RDP 访问连接到您的 EC2 Windows 实例。有关如何创建密钥对的信息，请参阅Windows 实例用户指南中的 Amazon EC2 密钥对和 Windows 实例。
- 至少有一个公有子网和一个私有子网以及一个安全组的 VPC。您可以使用您的原定设置 VPC。您不需要 NAT 网关或设备。AWS Directory Service 不支持 Active Directory 的网络地址转换（NAT）。为了实现这一点，您的 VPC 中必须有 Active Directory、FSx for Windows File Server 文件系统、ECS 集群和 EC2 实例。有关 VPC 和 Active Directory 的更多信息，请参阅 AWS 托管 Microsoft AD 先决条件中的 Amazon VPC 控制台向导配置。
- IAM ecsInstanceRole 和 ecsTaskExecutionRole 权限与您的账户关联。这些服务链接角色允许服务代表您进行 API 调用并访问容器、密钥、目录和文件服务器。

步骤 1：创建 IAM 访问角色

使用 AWS Management Console 创建集群

1. 请参阅 Amazon ECS 容器实例 IAM 角色 来检查你是否有ecsInstanceRole，如果没有，查看你如何创建。
2. 我们建议为实际生产环境中的最低权限自定义角色策略。为了完成本教程，请验证以下 AWS 托管策略附加到您的 ecsInstanceRole 上。如果尚未附加策略，则附加策略。
   - AmazonEC2ContainerServiceforEC2Role
   - AmazonSSMManagedInstanceCore

- AmazonSSMDirectoryServiceAccess

要附加 AWS 托管策略

a. 打开 IAM 控制台。
b. 在导航窗格中，选择 Roles (角色)。
c. 选择 AWS 托管角色。
d. 依次选择权限、附加策略。
e. 要缩小要附加的可用策略范围，请使用筛选条件。
f. 选择适当的策略，然后选择附加策略。

3. 请参阅 Amazon ECS 任务执行 IAM 角色 来检查你是否有ecsTaskExecutionRole，如果没有，查看你如何创建。

   我们建议为实际生产环境中的最低权限自定义角色策略。为了完成本教程，请验证以下 AWS 托管策略附加到您的 ecsTaskExecutionRole上。如果尚未附加策略，则附加策略。使用上一节中给出的过程附加 AWS 托管策略。

   - SecretsManagerReadWrite
   - AmazonFSxReadOnlyAccess
   - AmazonSSMReadOnlyAccess
   - AmazonECSTaskExecutionRolePolicy

步骤 2：创建 Windows Active Directory (AD)

1. 按照 AWS Directory Service 管理指南中创建 AWS 托管 AD 目录中描述的步骤进行操作。使用您为本教程指定的 VPC。在创建您的 AWS 托管式 AD Directory 的步骤 3 中,保存用户名和密码，以便在以下步骤中使用。另外，请注意完全限定的域名，以备将来使用。在创建 Active Directory 时，您可以继续完成以下步骤。
2. 创建要在以下步骤中使用的 AWS Secrets Manager 密码。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的 AWSSecrets Manager 入门。

   a. 打开 Secrets Manager 控制台。
   b. 单击存储新密钥。
   c. 选择其他密钥类型。
   d. 对于私有密钥/值，在第一行创建值为 **admin** 的密钥 **username**。点击+ 添加行。

e. 在新行中，创建密钥 **password**。对于值，键入在创建 AWS 托管 AD 目录的步骤 3 中输入的密码。

f. 点击下一步按钮。

g. 提供密钥名称和说明。单击下一步。

h. 单击下一步。单击存储。

i. 在密钥列表页面上，单击您刚刚创建的密钥。

j. 保存新密钥的 ARN，以便在以下步骤中使用。

k. 您的 Active Directory 正在创建时，您可以继续执行下一步。

步骤 3：确认并更新安全组规则

在此步骤中，您将验证并更新您正在使用的安全组的规则。对此，您可以使用为 VPC 创建的原定设置安全组。

验证并更新安全组。

您需要创建或编辑安全组以从端口发送数据和向端口发送数据，在 FSx for Windows File Server 用户指南中的 Amazon VPC 安全组中描述这些端口。您可以通过创建下表入站规则的第一行中显示的安全组入站规则来执行此操作。规则允许来自分配给相同安全组的网络接口（及其关联实例）的入站流量。您创建的所有云资源都位于同一 VPC 中，并连接到同一安全组。因此，此规则允许根据需要向 FSx for Windows File Server 系统、Active Directory 和 ECS 实例发送流量。其他入站规则允许流量为网站提供服务，并允许 RDP 访问连接到 ECS 实例。

下表显示了本教程需要哪些安全组入站规则。

类型	协议	端口范围	来源
所有流量	All	全部	*sg-securitygroup*
HTTPS	TCP	443	0.0.0.0/0
RDP	TCP	3389	您的笔记本电脑 IP 地址

下表显示了本教程所需的安全组出站规则。

类型	协议	端口范围	目标位置
所有流量	All	全部	0.0.0.0/0

1. 打开EC2 控制台并从左侧菜单中选择安全组。
2. 从现在显示的安全组列表中，选中用于本教程的安全组左侧的复选框。

   显示您的安全组详细信息。
3. 通过选择 Inbound rules（入站规则）或 Outbound rules（出站规则）选项卡并选择 Edit inbound rules（编辑入站规则）或 Edit outbound rules（编辑出站规则）按钮编辑入站和出站规则。编辑规则以匹配前表中显示的规则。在本教程稍后创建EC2实例后，使用 EC2 实例的公共IP地址编辑入站规则 RDP 源，如Amazon EC2 Windows 实例用户指南中的连接到Windows实例中所述。

步骤 4：创建 Amazon FSx for Windows File Server 文件系统

验证并更新安全组，创建 Active Directory 并处于活动状态后，在与 Active Directory 相同的 VPC 中创建 FSx for Windows File Server 文件系统。使用以下步骤创建 FSx for Windows File Server 文件系统供 Windows 任务使用。

创建您的第一个文件系统。

1. 打开 Amazon FSx 控制台。
2. 在控制面板上，选择创建文件系统以启动文件系统创建向导。
3. 在选择文件系统类型页面上，选择 FSx for Windows File Server，然后选择下一步。显示创建文件系统页面。
4. 在文件系统详细信息部分中，为您的文件系统提供一个名称。命名您的文件系统可以更轻松地查找和管理它们。最多可以使用 256 个 Unicode 字符。允许的字符包括字母、数字、空格和特殊字符加号 (+)。减号 (-)、等号 (=)、句点 (.)、下划线 (_)、冒号 (:) 和正斜杠 (/)。
5. 对于部署类型，选择单可用区部署部署在单个可用区中的文件系统。单可用区 2是最新一代的单可用区文件系统，支持 SSD 和 HDD 存储。
6. 对于存储类型，选择 HDD。
7. 对于存储容量，输入最小存储容量。
8. 保持吞吐量容量设置为原定设置。

9. 在网络和安全部分，选择您为 AWS Directory Service 目录选择的同一个 Amazon VPC。
10. 对于VPC 安全组，选择选择在步骤3：验证和更新安全组中验证的安全组。
11. 对于Windows 身份验证，选择 AWS托管 Microsoft Active Directory，然后选择您的 AWS Directory Service 目录。
12. 对于加密，请保留 aws/fsx (原定设置)的原定设置加密密钥设置。
13. 保留维护首选项的原定设置。
14. 点击下一步按钮。
15. 检查创建文件系统页面上显示的文件系统配置。请注意创建文件系统后可以修改的文件系统设置（供您参考）。选择创建文件系统。
16. 请记下文件系统 ID 值。您将在后面的步骤中用到它。

    在创建 FSx for Windows File Server 文件系统时，您可以继续执行后续步骤以创建集群和 EC2 实例。

步骤 5：创建 Amazon ECS 集群

使用 Amazon ECS 控制台创建集群

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在集群配置下方的集群名称中，输入 windows-fsx-cluster。
6. 展开基础设施，清除 AWS Fargate（无服务器），然后选择 Amazon EC2 实例。
   - 要创建 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择 Create new group（创建新组），然后提供有关组的以下详细信息：
     - 对于操作系统/架构，请选择 Windows Server 2019 Core。
     - 对于 EC2 实例类型，选择 t2.medium 或 t2.micro。
7. 选择创建。

步骤 6：创建经 Amazon ECS 优化的 Amazon EC2 实例

创建 Amazon ECS Windows 容器实例。

创建 Amazon ECS 实例

1. 使用 `aws ssm get-parameters` 命令检索托管您 VPC 的区域的 AMI 名称。有关更多信息，请参阅检索经 Amazon ECS 优化的 AMI 元数据。
2. 用来创建启动实例的 Amazon EC2 控制台。
   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
   b. 从导航栏中，选择要使用的区域。
   c. 从 EC2 控制面板中，选择 Launch Instance（启动实例）。
   d. 对于 Name (名称)，输入唯一的名称。
   e. 对于应用程序和操作系统映像（Amazon 系统映像），在搜索字段中，输入您检索的 AMI 名称。
   f. 对于实例类型，选择 t2.medium 或 t2.micro。
   g. 对于 Key pair (login)（密钥对 [登录]），选择一个密钥对。如果您未指定密钥对，则您
   h. 在网络设置下，对于 VPC 和子网，选择您的 VPC 和公有子网。
   i. 在 Network settings（网络设置）下，对于 Security group（安全组），可以选择现有安全组或创建新安全组。确保您选择的安全组在 教程的先决条件 中定义了入站和出站规则
   j. 在 Network settings（网络设置）下，对于 Auto-assign Public IP（自动分配公有 IP），选择 Enable（启用）。
   k. 展开高级详细信息，然后对于域加入目录，选择您创建的 Active Directory 的 ID。此选项域在 EC2 实例启动时加入您的 AD。
   l. 在 Advanced details（高级详细信息）下，对于 IAM instance profile（IAM 实例配置文件），选择 ecsInstanceRole。
   m. 使用以下用户数据配置您的 Amazon ECS 容器实例。在 Advanced Details（高级详细信息）下，将以下脚本粘贴到 User data（用户数据）字段中，将 *`cluster_name`* 替换为您的集群的名称。

      ```
      <powershell>
      Initialize-ECSAgent -Cluster windows-fsx-cluster -EnableTaskIAMRole
      </powershell>
      ```

   n. 准备好后，选中确认字段，然后选择 Launch Instances。
   o. 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。
3. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。

4. 在导航窗格中，选择集群，然后选择 windows-fsx-cluster。
5. 选择基础设施选项卡，验证您的实例是否已在 windows-fsx-cluster 集群中注册。

步骤 7：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。以下任务定义示例显示一个简单网页。此任务将启动两个具有 FSX 文件系统访问权限的容器。第一个容器将 HTML 文件写入文件系统。第二个容器从文件系统下载 HTML 文件并提供网页。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。
3. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
4. 在 JSON 编辑器框中，替换任务执行角色的值以及有关 FSx 文件系统的详细信息，然后选择保存。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\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>It
 Works!</h2> <p>You are using Amazon FSx for Windows File Server file system for
 persistent container storage.</p>' -Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
```

```
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force;
 Start-Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -
Destination C:\\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe
 w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
```

```
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

步骤 8：运行任务并查看结果

在运行任务之前，请验证 FSx for Windows File Server 文件系统的状态是Available。可用后，您可以使用创建的任务定义运行任务。任务首先是创建容器，使用文件系统在它们之间随机打开 HTML 文件。随机打开后，Web 服务器提供简单的 HTML 页面。

> **Note**
>
> 您可能无法从 VPN 内连接到网站。

使用 Amazon ECS 控制台运行任务并查看结果。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群，然后选择 windows-fsx-cluster。
3. 选择任务选项卡，然后选择运行新任务。
4. 对于启动类型，选择 EC2。
5. 在部署配置下的任务定义中，选择 fsx-windows，然后选择创建。
6. 当您的任务状态为 RUNNING 时，选择任务 ID。
7. 在容器下，当 container1 状态为 STOPPED 时，选择 container2 以查看容器的详细信息。
8. 在 container2 的容器详细信息下，选择网络绑定，然后单击与容器关联的外部 IP 地址。您的浏览器将打开并显示以下消息。

```
Amazon ECS Sample App
It Works!
You are using Amazon FSx for Windows File Server file system for persistent
 container storage.
```

> **Note**
>
> 显示消息可能需要几分钟时间。如果您在几分钟后未看到此消息，请检查是否未在 VPN 中运行，并确保容器实例的安全组允许端口 443 上的入站网络 HTTP 通信。

步骤 9：清除

> **Note**
>
> 要花费 20 到45 分钟删除 FSx for Windows File Server 文件系统或 AD。您必须等待，直到 FSx for Windows File Server 文件系统删除操作完成，然后再开始 AD 删除操作。

删除 FSx for Windows File Server 文件系统。

1. 打开 Amazon FSx 控制台。
2. 选择刚刚创建的 FSx for Windows File Server 文件系统左侧的单选按钮。
3. 选择操作。
4. 选择删除文件系统。

删除 AD。

1. 打开AWS Directory Service控制台。
2. 选择刚刚创建的 AD 左侧的单选按钮。
3. 选择操作。
4. 选择删除目录。

请删除集群。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群，然后选择 fsx-windows-cluster。
3. 选择删除集群。
4. 输入短语，然后选择删除。

终止 EC2 实例。

1. 打开 Amazon EC2 控制台。
2. 从左侧菜单中，选择实例。
3. 选中您创建的 EC2 实例左侧的框。
4. 单击实例状态、终止实例。

删除密钥。

1. 打开 Secrets Manager 控制台。
2. 选择您为本演练创建的密钥。
3. 点击操作。
4. 选择删除密钥。

将 Docker 卷与 Amazon ECS 结合使用

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如下：

- 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

使用 Docker 卷的注意事项

使用 Docker 卷时考虑以下事项：

- 只有在使用 EC2 启动类型或外部实例时，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，确保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：
  - 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI：

    ```
    sudo systemctl restart ecs
    ```

  - 对于经 Amazon ECS 优化的 Amazon Linux AMI：

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

在 Amazon ECS 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`。

以下任务定义 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。最多允许 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 对象的 sourceVolume 参数中引用。

dockerVolumeConfiguration

类型：DockerVolumeConfiguration 对象

必需：否

使用 Docker 卷时将指定此参数。只有在 EC2 实例上运行任务时，才支持 Docker 卷。Windows 容器仅支持使用 local 驱动程序。要使用绑定挂载，请改为指定 host。

scope

类型：字符串

有效值：task | shared

必需：否

Docker 卷的范围，可确定其生命周期。当任务开始时，将自动预配置范围限定为 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。

autoprovision

类型：布尔值

默认值：false

必需：否

如果此值为 `true`，则将创建 Docker 卷（如果此卷不存在）。仅在 `scope` 为 `shared` 时使用此字段。如果 `scope` 为 `task`，则必须省略此参数或将其设置为 `false`。

`driver`

类型：字符串

必需：否

要使用的 Docker 卷驱动程序。由于驱动程序值用于任务放置，因此，该名称必须与 Docker 提供的驱动程序名称匹配。如果已使用 Docker 插件 CLI 创建驱动程序，则使用 `docker plugin ls` 可从容器实例中检索驱动程序名称。如果已使用其他方法安装驱动程序，则使用 Docker 插件发现功能可检索驱动程序名称。有关更多信息，请参阅 Docker 插件发现。此参数映射到 Docker Remote API 的创建卷部分中的 `Driver` 以及 `docker volume create` 的 `--driver` 选项。

`driverOpts`

类型：字符串

必需：否

要传递的 Docker 驱动程序特定的选项的映射。此参数映射到 Docker Remote API 的创建卷部分中的 `DriverOpts` 以及 `docker volume create` 的 `--opt` 选项。

`labels`

类型：字符串

必需：否

要添加到 Docker 卷的自定义元数据。此参数映射到 Docker Remote API 的创建卷部分中的 `Labels` 以及 `docker volume create` 的 `--label` 选项。

`mountPoints`

类型：对象数组

必需：否

容器中数据卷的挂载点。此参数将映射到 Docker Remote API 的创建容器部分中的 `Volumes` 以及 docker run 的 `--volume` 选项。

Windows 容器可在 `$env:ProgramData` 所在的驱动器上挂载整个目录。Windows 容器无法在其他驱动器上挂载目录，并且挂载点不能跨驱动器使用。您必须指定挂载点才能将 Amazon EBS 卷直接附加到 Amazon ECS 任务。

`sourceVolume`

类型：字符串

必需：是，当使用 `mountPoints` 时

要挂载的卷的名称。

`containerPath`

类型：字符串

必需：是，当使用 `mountPoints` 时

挂载卷的容器中的路径。

`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`。

Docker 卷 ECS 示例

要使用 Docker 卷为容器提供临时存储

在此示例中，容器使用在任务完成后处置的空数据卷。一个示例使用案例为，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 范围以便卷持续存在，将自动预置设置为 true。这样就可以创建卷以供使用。然后，还可使用内置的 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"
        }
      ]
    }
  ]
```

使用 Docker 卷为容器提供 NFS 持久性存储

在此示例中，容器使用 NFS 数据卷，该数据卷在任务启动时自动挂载，在任务停止时自动卸载。这使用了 Docker 内置 local 驱动程序。一个示例使用案例为，您可能具有本地 NFS 存储，并且需要从 ECS Anywhere 任务访问该存储。可以通过带有 NFS 驱动程序选项的 Docker 卷实现此目标。

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 task 范围以便在任务停止后卸载卷。使用 local 驱动程序并使用 type、device 和 o 选项相应地配置 driverOpts。NFS_SERVER 替换为 NFS 服务器端点。

```
"volumes": [
       {
           "name": "NFS",
           "dockerVolumeConfiguration" : {
```

```
                "scope": "task",
                "driver": "local",
                "driverOpts": {
                    "type": "nfs",
                    "device": "$NFS_SERVER:/mnt/nfs",
                    "o": "addr=$NFS_SERVER"
                }
            }
        }
    ]
```

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": "NFS",
                  "containerPath": "/var/nfsmount"
                }
            ]
        }
    ]
```

将绑定挂载与 Amazon ECS 结合使用

使用绑定挂载，主机上的文件或目录（如 Amazon EC2 实例）被挂载到容器中。Fargate 和 Amazon EC2 实例上托管的任务都支持绑定挂载。绑定挂载与使用它们的容器的生命周期相关。停止使用绑定挂载的所有容器后（例如，停止任务时），数据将被删除。对于托管在 Amazon EC2 实例上的任务，可以通过在任务定义中指定 host 和可选值 sourcePath，将数据绑定到主机 Amazon EC2 实例的生命周期。有关更多信息，请参阅 Docker 文档中的使用绑定挂载。

以下是绑定挂载的常见使用案例。

- 提供空数据卷以挂载在一个或多个容器中。
- 在一个或多个容器中挂载主机数据卷。
- 与相同任务中的其他容器共享来自源容器的数据卷。
- 将 Dockerfile 路径及其内容公开到一个或多个容器。

使用绑定挂载时的注意事项

使用绑定挂载时应考虑以下事项。

- 默认情况下，使用平台版本 1.4.0 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）在 AWS Fargate 上托管的任务会收到至少 20GiB 的临时存储以用于绑定挂载。您可以增加临时存储总量，最多可达 200GiB，方法是在您的任务定义中指定 ephemeralStorage 参数。
- 要在任务运行时将 Dockerfile 中的文件公开到数据卷，Amazon ECS 数据平面将查找 VOLUME 指令。如果 VOLUME 指令中指定的绝对路径与任务定义中指定的 containerPath 绝对路径相同，则指令路径 VOLUME 中的数据将复制到数据卷。在下面的 Dockerfile 示例中，/var/log/exported 目录中名为 examplefile 的文件将写入主机，然后挂载到容器中。

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN mkdir -p /var/log/exported
  RUN touch /var/log/exported/examplefile
  VOLUME ["/var/log/exported"]
  ```

  预设情况下，卷权限设置为 0755 和所有者设置为 root。您可以在 Dockerfile 中自定义这些权限。以下示例将目录的所有者定义为 node。

  ```
  FROM public.ecr.aws/amazonlinux/amazonlinux:latest
  RUN yum install -y shadow-utils && yum clean all
  RUN useradd node
  RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
  RUN touch /var/log/exported/examplefile
  USER node
  VOLUME ["/var/log/exported"]
  ```

- 对于 Amazon EC2 实例上托管的任务，如果未指定 host 和 sourcePath 值，则 Docker 进程守护程序将为您管理绑定挂载。如果没有容器引用此绑定挂载，则 Amazon ECS 容器代理任务清理服务最终会将其删除。默认情况下，这发生在容器退出三个小时后。但是，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。如果需要在容器的生命周期内保持此数据，请为绑定挂载指定一个 sourcePath 值。

在 Amazon ECS 任务定义中指定绑定挂载

对于在 Fargate 或 Amazon EC2 实例上托管的 Amazon ECS 任务，以下任务定义 JSON 代码段显示任务定义的 volumes、mountPoints 和 ephemeralStorage 对象的语法。

```
{
   "family": "",
   ...
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
               "containerPath" : "/path/to/mount_volume",
               "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
          "name" : "string"
       }
    ],
    "ephemeralStorage": {
    "sizeInGiB": integer
    }
}
```

对于 Amazon EC2 实例上托管的 Amazon ECS 任务，您可以在指定任务卷详细信息时使用可选的 host 参数和 sourcePath。指定后，它将绑定挂载与任务的生命周期而不是容器关联起来。

```
"volumes" : [
    {
        "host" : {
            "sourcePath" : "string"
        },
        "name" : "string"
    }
]
```

下面将更详细地描述每个任务定义参数。

name

类型：字符串

必需：否

卷的名称。最多允许 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 对象的 sourceVolume 参数中引用。

host

必需：否

host 参数用于将绑定挂载的生命周期绑定到主机 Amazon EC2 实例（而不是任务）及其存储位置。如果 host 参数为空，则 Docker 进程守护程序将为您的数据卷分配一个主机路径，但不保证数据在与其关联的容器停止运行后将保留。

Windows 容器可在 $env:ProgramData 所在的驱动器上挂载整个目录。

> **Note**
>
> sourcePath 参数仅在使用 Amazon EC2 实例上托管的任务时才支持。

sourcePath

类型：字符串

必需：否

在使用 host 参数时，指定 sourcePath 可声明提供给容器的主机 Amazon EC2 容器实例上的路径。如果此参数为空，则 Docker 进程守护程序将为您分配一个主机路径。如果 host 参数包含 sourcePath 文件位置，则数据卷将在主机 Amazon EC2 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 Amazon EC2 容器实例上不存在 sourcePath 值，则 Docker 进程守护程序将创建该值。如果该位置不存在，则将导出源路径文件夹的内容。

mountPoints

类型：对象数组

必需：否

容器中数据卷的挂载点。此参数将映射到 Docker Remote API 的创建容器部分中的 Volumes 以及 docker run 的 --volume 选项。

Windows 容器可在 $env:ProgramData 所在的驱动器上挂载整个目录。Windows 容器无法在其他驱动器上挂载目录，并且挂载点不能跨驱动器使用。您必须指定挂载点才能将 Amazon EBS 卷直接附加到 Amazon ECS 任务。

sourceVolume

类型：字符串

必需：是，当使用 mountPoints 时

要挂载的卷的名称。

containerPath

类型：字符串

必需：是，当使用 mountPoints 时

挂载卷的容器中的路径。

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。

ephemeralStorage

类型：对象

必需：否

要为任务分配的临时存储容量。对于使用平台版本 1.4.0 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）的 AWS Fargate 上托管的任务，此参数用于将短暂可用存储的总量扩展到默认数量之外。

您可以使用 Copilot CLI、CloudFormation、AWS SDK 或 CLI 为绑定挂载指定临时存储容量。

绑定挂载示例

为 Fargate 任务分配更多的临时存储空间

对于在使用平台版本 1.4.0 或更高版本（Linux）或 1.0.0（Windows）的 Fargate 上托管的 Amazon ECS 任务，您可以为任务中要使用的容器分配超过原定设置数量的短暂存储。此示例可以合并到其他示例中，以便为 Fargate 任务分配更多的临时存储空间。

- 在任务定义中，定义 ephemeralStorage 对象。sizeInGiB 必须是介于 21 和 200 之间的整数，单位是 GiB。

```
"ephemeralStorage": {
    "sizeInGiB": integer
}
```

为一个或多个容器提供空数据卷

在某些用例中，您希望为任务中的容器提供一些临时空间。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。可以通过绑定挂载实现此目标。

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

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

2. 在 containerDefinitions 部分中，创建数据库容器定义。以便它们挂载卷。

```
"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"
        }
      ]
    }
  ]
```

将 Dockerfile 中的路径及其内容公开给容器

在本例中，您有一个 Dockerfile，用于写入要挂载到容器中的数据。此示例适用于 Fargate 或 Amazon EC2 实例上托管的任务。

1. 创建 Dockerfile。下面的示例使用公共 Amazon Linux2 容器映像，并在我们希望挂载容器的 `/var/log/exported` 目录中创建一个名为 `examplefile` 的文件。`VOLUME` 指令应该指定绝对路径。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN mkdir -p /var/log/exported
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

   预设情况下，卷权限设置为 `0755` 和所有者设置为 `root`。可以在 Dockerfile 中变更这些权限。在下面的示例中，`/var/log/exported` 目录的所有者设置为 `node`。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN yum install -y shadow-utils && yum clean all
   RUN useradd node
   RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
   USER node
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

2. 在任务定义 `volumes` 部分中，使用名称 `application_logs` 定义卷。

   ```
   "volumes": [
     {
       "name": "application_logs"
     }
   ```

```
]
```

3. 在 containerDefinitions 部分中，创建应用程序容器定义。以便它们挂载存储。containerPath 值必须与 Dockerfile 的 VOLUME 指令中指定的绝对路径匹配。

```
"containerDefinitions": [
  {
    "name": "application1",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  },
  {
    "name": "application2",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  }
]
```

为与主机 Amazon EC2 实例生命周期相关的容器提供空数据卷

对于在 Amazon EC2 实例上托管的任务，可以使用绑定挂载，并将数据绑定到主机 Amazon EC2 实例的生命周期。您可以通过使用 host 参数并指定 sourcePath 值进行此操作。存在于 sourcePath 的任何文件将提交至值为 containerPath 的容器。写入 containerPath 值的任何文件将会写入主机 Amazon EC2 实例上的 sourcePath 值。

> **Important**
>
> Amazon ECS 不会跨 Amazon EC2 实例同步您的存储。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 Amazon EC2 实例上。如果任务在停止和重新启动后需要持久存储，应始终在任务启动时使用 AWS CLI start-task 命令指定相同的 Amazon EC2 实例。您也可以将 Amazon EFS 卷用于持久性存储。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。

1. 在任务定义 `volumes` 部分中，使用 `name` 和 `sourcePath` 值定义绑定挂载。在下面的示例中，主机 Amazon EC2 实例包含 `/ecs/webdata` 您希望挂载容器中的数据。

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

2. 在 `containerDefinitions` 部分中，使用引用绑定挂载的名称的 `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"
```

```
      }
    ]
  }
]
```

在不同位置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，您的主机容器在 /data/webroot 有一个网站数据文件夹。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

对于托管在 Amazon EC2 实例上的任务，您可以在一个容器上定义一个或多个卷，然后在相同任务内的不同容器定义中使用 volumesFrom 参数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）要共享映像中内置的卷，请使用 Dockerfile 中的 VOLUME 指令。以下示例 Dockerfile 使用了 httpd 映像，然后添加卷并在 Apache 文档根目录中的 dockerfile_volume 处挂载卷。这是 httpd Web 服务器使用的文件夹。

   ```
   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 进程守护程序管理。还有一个定义的主机卷，被称为 host_etc。它在主机容器实例上导出 /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 容器会挂载 empty 和 host_etc 卷。该容器使用通过 Dockerfile 中的卷构建的映像。

```
"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 容器上的所有卷以同样方式挂载在 busybox 容器上。这包括在 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文件。此文件在每个卷文件夹中被称为 date。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。

管理 Amazon ECS 上的容器交换内存空间

使用 Amazon ECS，您可以在容器级别控制基于 Linux 的 Amazon EC2 实例上的交换内存空间使用量。使用每个容器的交换配置，任务定义中的每个容器都可以启用或禁用交换。对于启用此功能的用户，可以对所用的最大交换空间量进行限制。例如，延迟关键型容器可以禁用交换。相比之下，具有高暂时内存需求的容器可以打开交换，以减少容器负载时出现内存不足错误的机率。

容器的交换配置由以下容器定义参数管理。

maxSwap

容器可以使用的交换内存总量（以 MiB 为单位）。该参数会转换为 Docker 运行的--memory-swap选项，其中，该值为容器内存和maxSwap值之和。

如果指定maxSwap值为0，则该容器不使用交换。接受的值为0或任何正整数。如果省略 maxSwap 参数，该容器将为其运行所在的容器实例使用交换配置。必须为要使用的swappiness参数设置maxSwap值。

swappiness

您可以使用此功能调整容器的内存 swappiness 行为。swappiness 值为 0 不会导致交换，除非有需要。swappiness 值为 100 将导致页面被积极地交换。接受的值为0到100之间的整数。如果未指定swappiness参数，则使用默认值60。如果未指定 maxSwap 的值，则此参数将被忽略。此参数会将--memory-swappiness选项映射到 Docker 运行。

在以下示例中，提供了 JSON 语法。

```
"containerDefinitions": [{
        ...
        "linuxParameters": {
            "maxSwap": integer,
            "swappiness": integer
        },
        ...
```

```
}]
```

注意事项

在使用每个容器交换配置时，请考虑以下事项。

- 必须在托管任务的 Amazon EC2 实例上启用和分配交换空间，以供容器使用。预设情况下，经 Amazon ECS 优化的 AMI 没有启用交换功能。必须在实例上启用交换才能使用此功能。有关更多信息，请参阅《Amazon EC2 用户指南》的中的实例存储交换卷或如何使用交换文件分配内存以便在 Amazon EC2 实例中用作交换空间？。
- 仅指定 EC2 启动类型的任务定义支持交换空间容器定义参数。仅供 Fargate 上的 Amazon ECS 使用的任务定义不支持这些参数。
- 仅 Linux 容器支持此功能。目前不支持 Windows 容器。
- 如果任务定义中忽略了 `maxSwap` 和 `swappiness` 任务定义参数，每个容器都有默认的 `swappiness` 值 `60`。此外，总交换使用量限制为容器内存的两倍。
- 如果您在 Amazon Linux 2023 上使用任务，则不支持 `swappiness` 参数。

Fargate 启动类型的 Amazon ECS 任务定义差异

要使用 Fargate，您必须将任务定义配置为使用 Fargate 启动类型。使用 Fargate 时还有其他注意事项。

任务定义参数

使用 Fargate 启动类型的任务并非支持所有可用的 Amazon ECS 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`
- `placementConstraints`
- `privileged`
- `maxSwap`
- `swappiness`

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`，您可以添加的唯一功能是 `CAP_SYS_PTRACE`。`devices`、`sharedMemorySize` 和 `tmpfs` 参数不受支持。有关更多信息，请参阅 Linux 参数。
- `volumes` – Fargate 任务仅支持绑定挂载主机卷，因此， `dockerVolumeConfiguration` 参数不受支持。有关更多信息，请参阅 卷。
- `cpu` - 对于 AWS Fargate 上的 Windows 容器，该值不能小于 1 vCPU。

为确保您的任务定义可验证用于 Fargate，可在注册任务定义时指定以下内容：

- 在 AWS Management Console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。

操作系统和架构

为 AWS Fargate 配置任务和容器定义时，您必须指定容器运行的操作系统。AWS Fargate 支持以下操作系统：

- Amazon Linux 2

> **Note**
>
> Linux 容器仅使用主机操作系统中的内核和内核配置。例如，内核配置包括 `sysctl` 系统控件。Linux 容器映像可以由包含任何 Linux 发行版中的文件和程序的基本映像创建。如果 CPU 架构匹配，则可以在任何操作系统上通过任何 Linux 容器映像运行容器。

- Windows Server 2019 Full

- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

当您在 AWS Fargate 上运行 Windows 容器时，您必须具有 X86_64 CPU 架构。

当您在 AWS Fargate 上运行 Linux 容器时，您可以使用 X86_64 CPU 架构，也可以使用 ARM64 架构用于基于 ARM 的应用程序。有关更多信息，请参阅 the section called “适用于 64 位 ARM 工作负载的任务定义”。

任务 CPU 和内存

AWS Fargate 的 Amazon ECS 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。您可以通过以 MiB 或 GB 字符串方式在任务定义中指定内存值。例如，您可以使用 `3072`（以 MiB 为单位）或 `3 GB`（以 GB 为单位）指定内存值。您可以通过 CPU 单位数或虚拟 CPU（vCPU）数字符串的方式在 JSON 文件中将 CPU 值。例如，您可以使用 `1024`（CPU 单位数）或 `1 vCPU`（vCPU 数）来指定 CPU 值。

CPU 值	内存值	AWS Fargate 支持的操作系统
256 (.25 vCPU)	512MiB、1GB、2GB	Linux
512 (.5 vCPU)	1GB、2GB、3GB、4GB	Linux
1024 (1 vCPU)	2GB、3GB、4GB、5GB、6GB、7GB、8GB	Linux、Windows
2048 (2 vCPU)	4GB 到 16GB 之间 (以 1GB 为增量)	Linux、Windows
4096 (4 vCPU)	8GB 到 30GB 之间 (以 1GB 为增量)	Linux、Windows
8192 (8 vCPU)	16 GB 到 60 GB 之间（以 4 GB 为增量）	Linux

CPU 值	内存值	AWS Fargate 支持的操作系统
Note 此选项需要 Linux 平台 1.4.0 或更高版本。		
16384 (16vCPU) Note 此选项需要 Linux 平台 1.4.0 或更高版本。	32 GB 到 120 GB 之间（以 8 GB 为增量）	Linux

任务联网

AWS Fargate 的 Amazon ECS 任务要求使用 awsvpc 网络模式，该模式可为每个任务提供一个弹性网络接口。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，还需要一个到互联网的路由，或一个可以将请求路由到互联网的 NAT 网关。对于私有子网中拉取容器映像的 Fargate 任务，您需要私有子网中的 NAT 网关以将请求路由到互联网。在 Amazon ECR 中托管容器映像时，您可以将 Amazon ECR 配置为使用接口 VPC 端点。在这种情况下，任务的专用 IPv4 地址将用于图像提取。有关 Amazon ECR 接口端点的更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的 Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。

以下是 Fargate 服务的 networkConfiguration 部分的示例：

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

任务资源限制

AWS Fargate 上的 Linux 容器的 Amazon ECS 任务定义支持使用 `ulimits` 参数来定义要为容器设置的资源限制。

AWS Fargate 上的 Windows 的 Amazon ECS 任务定义支持使用 `ulimits` 参数来定义要为容器设置的资源限制。

Fargate 上托管的 Amazon ECS 任务使用操作系统设置的默认资源限制值，`nofile` 资源限制参数除外。`nofile` 资源限制对容器可以使用的打开文件数量设置限制。在 Fargate 上，默认的 `nofile` 软限制为 `1024`，硬限制为 `65535`。您可以将两个限制的值最高设置为 `1048576`。

下面是一个示例任务定义代码段，介绍如何定义 `nofile` 限制已经增加了一倍：

```
"ulimits": [
    {
       "name": "nofile",
       "softLimit": 2048,
       "hardLimit": 8192
    }
]
```

有关可以调整的其他资源限制的更多信息，请参阅资源限制。

日志记录

事件日志记录

Amazon ECS 将其采取的操作记录到 EventBridge 中。您可以使用 EventBridge 的 Amazon ECS 事件来接收有关 Amazon ECS 集群、服务和任务当前状态的近实时通知。此外，您还可以自动执行操作以响应这些事件。有关更多信息，请参阅 使用 EventBridge 自动响应 Amazon ECS 错误。

任务生命周期日志记录

在 Fargate 上运行的任务会发布时间戳，以便在任务生命周期的各个状态中跟踪任务。通过描述 AWS CLI 和开发工具包中的任务，可以在 AWS Management Console 中的任务详细信息中查看时间戳。例如，您可以使用时间戳来评估任务下载容器映像所花费的时间，并决定是应优化容器映像大小，还是使用 Seekable OCI 索引。有关容器映像实践的更多信息，请参阅 Amazon ECS 容器映像的最佳实践。

应用程序日志记录

AWS Fargate 的 Amazon ECS 任务定义支持用于日志配置的 awslogs、splunk 和 awsfirelens 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 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 将 Amazon ECS 日志发送到 CloudWatch 。

有关任务定义中的 awsfirelens 日志驱动程序的更多信息，请参阅将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。

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splunk 日志驱动程序。

任务存储

对于 Fargate 上托管的 Amazon ECS 任务，支持以下存储类型：

- Amazon EBS 卷为数据密集型容器化工作负载提供经济高效、持久、高性能的块存储。有关更多信息，请参阅 将 Amazon EBS 卷与 Amazon ECS 结合使用。
- Amazon EFS 卷（用于持久性存储）。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。
- 绑定挂载以实现临时存储。有关更多信息，请参阅 将绑定挂载与 Amazon ECS 结合使用。

使用 Seekable OCI（SOCI）延迟加载容器映像

Fargate 上使用 Linux 平台版本 1.4.0 的 Amazon ECS 任务可以使用 Seekable OCI（SOCI）来帮助更快地启动任务。借助 SOCI，容器只需花几秒钟的时间进行映像拉取即可启动，从而在后台下载映像时为环境设置和应用程序实例化提供了时间。这称为延迟加载。当 Fargate 启动 Amazon ECS 任务

时，Fargate 会自动检测任务中是否存在映像的 SOCI 索引，并在不等待下载整个映像的情况下启动容器。

对于在没有 SOCI 索引的情况下运行的容器，容器映像将在容器启动之前完全下载。此行为在 Fargate 的所有其他平台版本上以及 Amazon EC2 实例上经 Amazon ECS 优化的 AMI 上均相同。

Seekable OCI 索引

Seekable OCI（SOCI）是 AWS 开发的一种开源技术，它可以通过延迟加载容器映像来更快地启动容器。SOCI 的工作原理是为现有容器映像中的文件创建索引（SOCI 索引）。此索引有助于更快地启动容器，从而提供在下载整个映像之前从容器映像中提取单个文件的功能。SOCI 索引必须作为构件存储在与容器注册表中的映像相同的存储库中。您只能使用来自可信来源的 SOCI 索引，因为该索引是映像内容的权威来源。有关更多信息，请参阅介绍用于延迟加载容器映像的 Seekable OCI。

注意事项

如果您希望 Fargate 使用 SOCI 索引在任务中延迟加载容器映像，请考虑以下几点：

- 只有在 Linux 平台版本 `1.4.0` 上运行的任务才能使用 SOCI 索引。不支持在 Fargate 上运行 Windows 容器的任务。
- 支持在 X86_64 或 ARM64 CPU 架构上运行的任务。使用 ARM64 架构的 Linux 任务不支持 Fargate Spot 容量提供程序。
- 任务定义中的容器映像必须在与映像相同的容器注册表中具有 SOCI 索引。
- 任务定义中的容器映像必须存储在兼容的映像注册表中。以下列出了兼容的注册表：
  - Amazon ECR 私有注册表。
- 仅支持使用 gzip 压缩或未压缩的容器映像。不支持使用 zstd 压缩的容器映像。
- 建议您尝试使用大于 250 MiB 压缩后的容器映像进行延迟加载。您不太可能看到加载较小映像的时间缩短。
- 由于延迟加载可能会改变任务启动所需的时间，因此您可能需要对各种超时时间进行更改，例如 Elastic Load Balancing 的运行状况检查宽限期。
- 如果要防止延迟加载容器映像，请从容器注册表中删除 SOCI 索引。如果任务中的容器映像不符合其中一个注意事项，则使用默认方法下载该容器映像。

创建 Seekable OCI 索引

对于要延迟加载的容器映像，需要创建 SOCI 索引（元数据文件），并将其与容器映像一起存储在容器映像存储库中。要创建和推送 SOCI 索引，您可以使用 GitHub 上的开源 soci-snapshotter CLI 工具。

或者，您可以部署 CloudFormation AWS SOCI 索引构建器。这是一种无服务器解决方案，可以在将容器映像推送到 Amazon ECR 时自动创建和推送 SOCI 索引。有关解决方案和安装步骤的更多信息，请参阅 GitHub 上的 CloudFormation AWS SOCI 索引构建器。CloudFormation AWS SOCI 索引构建器是自动开始使用 SOCI 的一种方法，而开源 SOCI 工具在索引生成方面具有更大的灵活性，并且能够将索引生成集成到持续集成和持续交付（CI/CD）管道中。

> **Note**
>
> 要为映像创建 SOCI 索引，该映像必须存在于正在运行 `soci-snapshotter` 的计算机上的 containerd 映像存储中。如果映像在 Docker 映像存储中，则无法找到该映像。

验证任务是否使用延迟加载

要验证任务是否使用 SOCI 延迟加载，请从任务内部检查任务元数据端点。在您查询任务元数据端点版本 4 时，在您要查询的容器的默认路径中有一个 `Snapshotter` 字段。此外，`/task` 路径中每个容器都有 `Snapshotter` 字段。此字段的默认值为 `overlayfs`，如果使用 SOCI，则此字段设置为 `soci`。

运行 Windows 的 EC2 实例的 Amazon ECS 任务定义差异

在 EC2 Windows 实例上运行的任务并不支持所有可用的 Amazon ECS 任务定义参数。某些参数完全不受支持，而其他参数的行为则不同。

Amazon EC2 Windows 任务定义不支持以下任务定义参数：

- `containerDefinitions`
  - `disableNetworking`
  - `dnsServers`
  - `dnsSearchDomains`
  - `extraHosts`
  - `links`
  - `linuxParameters`
  - `privileged`
  - `readonlyRootFilesystem`
  - `user`
  - `ulimits`

- volumes
  - dockerVolumeConfiguration
- cpu

  我们建议为 Windows 容器指定容器级 CPU。
- memory

  我们建议为 Windows 容器指定容器级内存。
- proxyConfiguration
- ipcMode
- pidMode
- taskRoleArn

  EC2 Windows 实例上的任务的 IAM 角色需要额外配置，但此配置很大程度上类似于在 Linux 容器实例上配置任务的 IAM 角色。有关更多信息，请参阅the section called “ Amazon EC2 Windows 实例附加配置”。

使用控制台创建 Amazon ECS 任务定义

您可以通过使用控制台或编辑 JSON 文件来创建任务定义。

JSON 验证

Amazon ECS 控制台 JSON 编辑器会在 JSON 文件中验证以下各项：

- 该文件是有效的 JSON 文件。
- 该文件不包含任何无关的键。
- 该文件包含 familyName 参数。
- containerDefinitions 下方至少有一个条目。

AWS CloudFormation 堆栈

以下行为适用于 2023 年 1 月 12 日之前在新的 Amazon ECS 控制台中创建的任务定义。

当您创建任务定义时，Amazon ECS 控制台会自动创建一个名称以 ECS-Console-V2-TaskDefinition- 开头的 CloudFormation 堆栈。如果您使用 AWS CLI 或 AWS SDK 取消注册任务

定义，则必须手动删除任务定义堆栈。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的删除堆栈。

2023 年 1 月 12 日之后创建的任务定义不会为其自动创建 CloudFormation 堆栈。

过程

Amazon ECS console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。
3. 在创建新任务定义菜单上，选择创建新任务定义。
4. 对于Task definition family（任务定义系列）中，为任务定义指定唯一名称。
5. 对于启动类型，选择应用程序环境。控制台默认值为 AWS Fargate（无服务器）。Amazon ECS 使用此值执行验证，以确保任务定义参数对基础设施类型有效。
6. 对于操作系统/架构，为任务选择操作系统和 CPU 架构。

   要在 64 位 ARM 架构上运行任务，请选择 Linux/ARM64。有关更多信息，请参阅 the section called “运行时平台”。

   要在 Windows 容器上运行您的 AWS Fargate 任务，请选择受支持的 Windows 操作系统。有关更多信息，请参阅 the section called “操作系统和架构”。
7. 对于 Task size（任务大小），选择要为任务预留的 CPU 和内存值。CPU 值指定为 vCPU，内存指定为 GB。

   对于 Fargate 上托管的任务，下表显示了有效的 CPU 和内存组合。

CPU 值	内存值	AWS Fargate 支持的操作系统
256 (.25 vCPU)	512MiB、1GB、2GB	Linux
512 (.5 vCPU)	1GB、2GB、3GB、4GB	Linux
1024 (1 vCPU)	2GB、3GB、4 GB、5GB、6G B、7GB、8GB	Linux、Windows

CPU 值	内存值	AWS Fargate 支持的操作系统
2048 (2 vCPU)	4GB 到 16GB 之间 (以 1GB 为增量)	Linux、Windows
4096 (4 vCPU)	8GB 到 30GB 之间 (以 1GB 为增量)	Linux、Windows
8192 (8 vCPU) **Note** 此选项需要 Linux 平台 1.4.0 或更高版本。	16 GB 到 60 GB 之间（以 4 GB 为增量）	Linux
16384 (16vCPU) **Note** 此选项需要 Linux 平台 1.4.0 或更高版本。	32 GB 到 120 GB 之间（以 8 GB 为增量）	Linux

对于 Amazon EC2 上托管的任务，受支持的任务 CPU 值介于 128 个 CPU 单元（0.125 个 vCPU）和 10240 个 CPU 单元（10 个 vCPU）之间。要以 GB 为单位指定内存值，请在值的后面输入 GB。例如，要将内存值设置为 3GB，请输入 3GB。

> **Note**
>
> Windows 容器将忽略任务级 CPU 和内存参数。

8. 对于 Network mode（网络模式），选择要使用的网络模式。默认值为 awsvpc 模式。有关更多信息，请参阅 Amazon ECS 任务联网。

   如果您为主机端口在端口映射下选择桥接，请输入要为容器预留的容器实例上的端口号。

9. （可选）展开任务角色部分以配置任务的 AWS Identity and Access Management（IAM）角色：

   a. 对于 Task role（任务角色），选择要分配到任务的 IAM 角色。任务 IAM 角色为要调用 AWS API 操作的任务中的容器提供权限。

   b. 对于任务执行角色，请选择角色。

      有关何时使用任务执行角色的信息，请参阅 the section called “任务执行 IAM 角色”。如果您不需要该角色，请选择无。

10. 对于在您的任务定义中定义的每个容器，请完成以下步骤。

    a. 对于 Name（名称），输入容器的名称。

    b. 对于 Image URI（映像 URI），输入用于启动容器的映像。Amazon ECR 公开映像浏览馆注册表中的映像只能使用 Amazon ECR 公开注册表名称来指定。例如，如果 `public.ecr.aws/ecs/amazon-ecs-agent:latest` 已指定，则使用 Amazon ECR 公开映像浏览馆上托管的 Amazon Linux 容器。对于所有其他存储库，请使用 `repository-url/image:tag` 或 `repository-url/image@digest` 格式指定存储库。

    c. 如果您的映像位于 Amazon ECR 之外的私有注册表中，请在私有注册表下开启私有注册表身份验证。然后，在 Secrets Manager ARN 或名称中，输入密钥的 Amazon 资源名称（ARN）。

    d. 对于必需容器，如果您的任务定义中定义了两个或更多容器，则可以指定是否应将该容器视为必需容器。当一个容器被标记为必需时，则在该容器停止时任务将停止。每个任务定义都必须至少包含一个关键容器。

    e. 端口映射可让容器访问主机容器上的端口以发送或接收流量。在 Port mappings（端口映射）下，请执行以下操作之一：

       - 当您使用 awsvpc 网络模式时，对于 Container port（容器端口）和 Protocol（协议），选择要用于容器的端口映射。
       - 当您使用 bridge 网络模式时，对于 Container port（容器端口）和 Protocol（协议），选择要用于容器的端口映射。

       选择 Add more port mappings（添加更多端口映射）以指定其他容器端口映射。

    f. 要为容器提供对其根文件系统的只读访问权，对于只读根文件系统，选择只读。

    g. （可选）要在资源分配限制下定义与任务级值不同的容器级 CPU、GPU 和内存限制，请执行以下操作：

- 对于 CPU，输入 Amazon ECS 容器代理为容器预留的 CPU 单元数。
- 对于 GPU，输入容器实例的 GPU 单元数。

  支持 GPU 的 Amazon EC2 实例为每个 GPU 配备 1 个 GPU 单元。有关更多信息，请参阅 the section called “适用于 GPU 工作负载的任务定义”。
- 对于内存硬限制，输入要提供给容器的内存量（以 GB 为单位）。如果容器试图超出硬限制，则容器将停止。
- Docker 20.10.0 或更高版本进程守护程序将为容器预留最少 6 兆字节（MiB）的内存，因此，请不要为容器指定 6MiB 以下的内存。

  Docker 19.03.13-ce 或较早版本进程守护程序将为容器预留最少 4MiB 的内存，因此，请不要为容器指定 4MiB 以下的内存。
- 对于内存软限制，输入要为容器预留的内存软限制（以 GB 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范围内。如果您未指定任务级内存，则必须为内存硬限制和内存软限制中的一个或两个指定非零整数。如果同时指定两者，则内存硬限制必须大于内存软限制。

  Windows 容器不支持此功能。

h. （可选）展开环境变量部分来指定要注入到容器中的环境变量。您可以使用键值对单独指定环境变量，也可以通过指定 Amazon S3 存储桶中托管的环境变量文件来批量指定环境变量。有关如何设置环境变量文件的格式的信息，请参阅将单个环境变量传递给 Amazon ECS 容器。

   当您为密钥存储指定环境变量时，请在密钥中输入密钥名称。然后，在 ValueFrom 中，输入 Systems Manager Parameter Store 密钥或 Secrets Manager 密钥的完整 ARN

i. （可选）选择 Use log collection（使用日志收集）选项来指定日志配置。对于每个可用的日志驱动程序，都有日志驱动程序选项要指定。默认选项将容器日志发送到 Amazon CloudWatch Logs。其他日志驱动程序选项都使用 AWS FireLens 进行配置。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。

   下面更详细地介绍了每个容器日志目标。

   - Amazon CloudWatch – 将任务配置为将容器日志发送到 CloudWatch Logs。提供了默认的日志驱动程序选项，用于代表您创建 CloudWatch 日志组。要指定其他日志组名称，请更改驱动程序选项值。

- 将日志导出到 Splunk – 将任务配置为将容器日志发送到将日志发送到远程服务的 Splunk 驱动程序。您必须输入 Splunk Web 服务的 URL。Splunk 令牌指定为密钥选项，因为它可能被视为敏感数据。
- 将日志导出到 Amazon Data Firehose – 将任务配置为将容器日志发送到 Firehose。提供了默认的日志驱动程序选项，该选项将日志发送到 Firehose 传输流。要指定其他传输流名称，请更改驱动程序选项值。
- 将日志导出到 Amazon Kinesis Data Streams – 将任务配置为将容器日志发送到 Kinesis Data Streams。提供了默认的日志驱动程序选项，该选项将日志发送到 Kinesis Data Streams 流。要指定其他传输流名称，请更改驱动程序选项值。
- 将日志导出到 Amazon OpenSearch Service – 将任务配置为将容器日志发送到 OpenSearch Service 域。必须提供日志驱动程序选项。
- 将日志导出到 Amazon S3 – 将任务配置为将容器日志发送到 Amazon S3 存储桶。提供了默认的日志驱动程序选项，但您必须指定有效的 Amazon S3 存储桶名称。

j. （可选）配置其他容器参数。

要配置此选项	请执行该操作	
Healthcheck 这些命令用于确定容器是否运行正常。有关更多信息，请参阅 使用容器运行状况检查确定 Amazon ECS 任务运行状况。	展开 HealthCheck，然后配置以下项目： • 对于 Command（命令），输入逗号分隔的命令列表。您可以设置命令以 CMD 开头，以直接运行命令参数，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，将使用 CMD。 • 对于 Interval（间隔），输入每两次运行状况检查之间的秒数。有效值为 5 到 30。 • 对于 Timeout（超时），输入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。有效值为 2 到 60。 • 对于 Start period（启动期间），输入运行状况检查命令运行之前，等待容器引导的时间长度（以秒为单位）。有效值为 0 到 300。 •	

要配置此选项	请执行该操作	
	对于 Retries（重试次数），输入出现故障时重试运行状况检查命令的次数。有效值为 1 到 10。	
容器超时 这些选项决定何时启动和停止容器。	展开容器超时，然后配置以下内容： • 要配置在放弃解析容器的依赖项之前等待的时间，请在开始超时时间中输入秒数。 • 要配置在容器自身没有正常退出的情况下停止之前等待的时间，请在停止超时时间中输入秒数。	

要配置此选项	请执行该操作	
容器网络设置 这些选项决定是否在容器内使用联网。	展开容器网络设置，然后配置以下内容： • 要禁用容器联网，请选择关闭联网。 • 要配置提供给容器的 DNS 服务器 IP 地址，请在 DNS 服务器中以单独的行输入每台服务器的 IP 地址。 • 要将 DNS 域配置为搜索提供给容器的非完全限定主机名，请在 DNS 搜索域中以单独的行输入每个域。 模式是 **^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$** 。 • 要配置容器主机名，请在主机名中输入容器 goat 名称。 • 要添加附加到容器上的 `/etc/hosts` 文件的主机名和 IP 地址映射，请选择添加额外主机，然后在主机名和 IP 地址中输入主机名和 IP 地址。	

要配置此选项	请执行该操作	
Docker 配置 它们会覆盖 Dockerfile 中的值。	展开 Docker 配置，然后配置以下项目： • 对于命令，输入容器的可执行命令。 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 选项。此参数会覆盖 Dockerfile 中的 CMD 指令。 • 对于入口点，输入传递给容器的 Docker ENTRYPOINT。 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。此参数会覆盖 Dockerfile 中的 ENTRYPOINT 指令。 • 对于工作目录，输入容器将运行所提供的任何入口点和命令指令的目录。 此参数将映射到 Docker Remote API 的创建容器部分中的 WorkingDi	

要配置此选项	请执行该操作	
	r 以及 docker run 的 --workdir 选项。此参数会覆盖 Dockerfile 中的 WORKDIR 指令。	
Ulimits 这些将覆盖操作系统的默认资源配额设置。 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。	展开资源限制（ulimits），然后选择添加 ulimit。对于限制名称，选择限制。然后，在软限制和硬限制中输入值。 要添加其他 ulimits，请选择添加 ulimit。	
Docker 标签 此选项将元数据添加到您的容器中。 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。	展开 Docker 标签，选择添加键值对，然后输入键和值。 要添加其他 Docker 标签，请选择添加键值对。	

要配置此选项	请执行该操作	
容器启动顺序 此选项定义容器启动和关闭的依赖项。一个容器可以包含多个依赖项。	展开启动依赖项排序，然后配置以下内容： a. 选择添加容器依赖项。 b. 对于容器，选择容器。 c. 在条件中，选择启动依赖项条件。 要添加其他依赖项，请选择添加容器依赖项。	

k. （可选）选择 Add more containers（添加更多容器）将其他容器添加到任务定义。

11. （可选）存储部分用于扩展 Fargate 上托管的任务的短暂存储量。您还可以使用此部分为任务添加数据卷配置。

    - 要将可用的短暂存储扩展到超出您的 Fargate 任务的默认值 20 gibibytes (GiB)，请在 Amount（量）中输入一个最高为 200 GiB 的值。

12. （可选）要为任务定义添加数据卷配置，选择添加卷，然后按照以下步骤操作。

    a. 对于 Volume name（卷名称），输入数据卷的名称。创建容器挂载点时，将使用数据卷名称。

    b. 对于卷配置，请选择是要在创建任务定义时还是在部署期间配置卷。

    > **Note**
    >
    > 可以在创建任务定义时配置的卷包括绑定挂载、Docker、Amazon EFS 和适用于 Windows File Server 的 Amazon FSx。运行任务或创建或更新服务时可以在部署时配置的卷包括 Amazon EBS。

    c. 对于卷类型，选择与所选配置类型兼容的卷类型，然后再配置该卷类型。

卷类型	步骤	
绑定挂载	a. 选择 Add mount point（添加挂载点），然后配置以下内容： • 对于 Container（容器），选择挂载点的容器。 • 对于 Source volume（源卷），选择要挂载到容器的数据卷。 • 对于 Container path（容器路径），输入挂载卷的容器上的路径。 • 对于只读，选择容器是否具有对卷的只读访问权。 b. 要添加其他挂载点，请选择 Add mount point（添加挂载点）。	

卷类型	步骤	
EFS	a. 对于 File system ID（文件系统 ID），选择 Amazon EFS 文件系统 ID。 b. （可选）对于 Root directory（根目录），输入 Amazon EFS 文件系统中要作为主机内的根目录挂载的目录。如果忽略此参数，将使用 Amazon EFS 卷的根目录。 如果您计划使用 EFS 接入点，请将此字段留为空白。 c. （可选）对于 Access point（接入点），选择要使用的接入点 ID。 d. （可选）要加密 Amazon EFS 文件系统和 Amazon ECS 主机之间的数据或在挂载卷时使用任务执行角色，请选择 Advanced configurations（高级配置），然后配置以下内容： • 要加密 Amazon EFS 文件系统和 Amazon ECS 主机之间的数据，请选择 Transit	

卷类型	步骤	
	encryption（传输加密），然后选择 Port（端口），输入在 Amazon ECS 主机和 Amazon EFS 服务器之间发送加密数据时使用的端口。如果未指定传输加密端口，将使用 Amazon EFS 挂载帮助程序使用的端口选择策略。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 EFS 挂载帮助程序。 • 要在挂载 Amazon EFS 文件系统时使用任务定义中定义的 Amazon ECS 任务 IAM 角色，请选择 IAM authorization（IAM 授权）。 e. 选择 Add mount point（添加挂载点），然后配置以下内容： • 对于 Container（容器），选择挂载点的容器。 • 对于 Source volume（源卷），选择要挂载到容器的数据卷。	

卷类型	步骤	
	• 对于 Container path（容器路径），输入挂载卷的容器上的路径。 • 对于只读，选择容器是否具有对卷的只读访问权。 f. 要添加其他挂载点，请选择 Add mount point（添加挂载点）。	

卷类型	步骤	
Docker	a. 对于驱动程序，输入 Docker 卷配置。Windows 容器仅支持使用本地驱动程序。要使用绑定挂载，请指定主机。 b. 对于 Scope（范围），选择卷生命周期。 • 要在任务开始和停止时保持生命周期，请选择 Task（任务）。 • 要在任务停止后保持卷，请选择 Shared（共享）。 c. 选择 Add mount point（添加挂载点），然后配置以下内容： • 对于 Container（容器），选择挂载点的容器。 • 对于 Source volume（源卷），选择要挂载到容器的数据卷。 • 对于 Container path（容器路径），输入挂载卷的容器上的路径。	

卷类型	步骤	
	• 对于只读，选择容器是否具有对卷的只读访问权。 d. 要添加其他挂载点，请选择 Add mount point（添加挂载点）。	

卷类型	步骤	
FSx for Windows File Server	a. 对于文件系统 ID，选择 FSx for Windows File Server 文件系统 ID。 b. 对于根目录，输入 FSx for Windows File Server 文件系统中要作为主机内的根目录挂载的目录。 c. 对于凭证参数，选择凭证的存储方式。 • 要使用 AWS Secrets Manager，请输入 Secrets Manager 密钥的 Amazon 资源名称（ARN）。 • 要使用 AWS Systems Manager，请输入 Systems Manager 参数的 Amazon 资源名称（ARN）。 d. 对于域，请输入由 AWS Directory Service for Microsoft Active Directory（）AWS Managed Microsoft AD）目录托管的完全限定域名或自托管 EC2 Active Directory。 e.	

卷类型	步骤	
	选择 Add mount point（添加挂载点），然后配置以下内容： • 对于 Container（容器），选择挂载点的容器。 • 对于 Source volume（源卷），选择要挂载到容器的数据卷。 • 对于 Container path（容器路径），输入挂载卷的容器上的路径。 • 对于只读，选择容器是否具有对卷的只读访问权。 f. 要添加其他挂载点，请选择 Add mount point（添加挂载点）。	

卷类型	步骤	
Amazon EBS	a. 选择 Add mount point（添加挂载点），然后配置以下内容： • 对于 Container（容器），选择挂载点的容器。 • 对于 Source volume（源卷），选择要挂载到容器的数据卷。 • 对于 Container path（容器路径），输入挂载卷的容器上的路径。 • 对于只读，选择容器是否具有对卷的只读访问权。 b. 要添加其他挂载点，请选择 Add mount point（添加挂载点）。	

13. 要从另一个容器中添加卷，请选择从中添加卷，然后配置以下内容：
    - 对于容器，选择容器。
    - 对于来源，选择包含要装载的卷的容器。
    - 对于只读，选择容器是否具有对卷的只读访问权。
14. （可选）要使用 AWS Distro for OpenTelemetry 集成配置应用程序跟踪和指标收集设置，请展开监控，然后选择使用指标收集来收集任务的指标并将其发送到或 Amazon CloudWatch

或 Amazon Managed Service for Prometheus。选择此选项后，Amazon ECS 会创建 AWS Distro for OpenTelemetry 容器附加，该附加已预先配置为发送应用程序指标。有关更多信息，请参阅 使用应用程序指标关联 Amazon ECS 应用程序性能。

a. 选中 Amazon CloudWatch 后，您的自定义应用程序指标将作为自定义指标路由到 CloudWatch。有关更多信息，请参阅 将应用程序指标导出到 Amazon CloudWatch。

> **Important**
>
> 将应用程序指标导出到 Amazon CloudWatch 时，您的任务定义需要具有所需权限的任务 IAM 角色。有关更多信息，请参阅 AWS Distro for OpenTelemetry 与 Amazon CloudWatch 集成所需的 IAM 权限。

b. 当您选择 Amazon Managed Service for Prometheus (Prometheus libraries instrumentation)（Amazon Managed Service for Prometheus（Prometheus 库分析））时，您的任务级 CPU、内存、网络和存储指标以及自定义应用程序指标都将路由到 Amazon Managed Service for Prometheus。对于工作空间远程写入端点，为 Prometheus 工作空间输入远程写入端点 URL。对于抓取目标，输入 AWS Distro for OpenTelemetry 收集器可用来抓取指标数据的主机和端口。有关更多信息，请参阅 将应用程序指标导出到 Amazon Managed Service for Prometheus。

> **Important**
>
> 将应用程序指标导出到 Amazon Managed Service for Prometheus 时，您的任务定义需要具有所需权限的任务 IAM 角色。有关更多信息，请参阅 AWS Distro for OpenTelemetry 与 Amazon Managed Service for Prometheus 集成所需的 IAM 权限。

c. 当您选择 Amazon Managed Service for Prometheus（OpenTelemetry 分析）时，您的任务级 CPU、内存、网络和存储指标以及自定义应用程序指标都将路由到 Amazon Managed Service for Prometheus。对于工作空间远程写入端点，为 Prometheus 工作空间输入远程写入端点 URL。有关更多信息，请参阅 将应用程序指标导出到 Amazon Managed Service for Prometheus。

> **Important**
>
> 将应用程序指标导出到 Amazon Managed Service for Prometheus 时，您的任务定义需要具有所需权限的任务 IAM 角色。有关更多信息，请参阅 AWS Distro for

OpenTelemetry 与 Amazon Managed Service for Prometheus 集成所需的 IAM 权限。

15. （可选）展开 Tags（标签）部分将标签（作为键值对）添加到任务定义中。
    - [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
      - 对于 Key（键），输入键名称。
      - 对于值，输入键值。
    - [删除标签] 在标签旁，选择 Remove tag (删除标签)。
16. 选择创建以注册任务定义。

Amazon ECS console JSON editor

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。
3. 在创建新的任务定义菜单上，选择使用 JSON 创建新的任务定义。
4. 在 JSON 编辑器框中，编辑您的 JSON 文件，

   JSON 必须通过 the section called “JSON 验证” 中指定的验证检查。
5. 选择创建。

使用控制台更新 Amazon ECS 任务定义

一个任务定义修订是当前任务定义的副本，新的参数值将替换现有参数值。您未修改的所有参数都在新修订版中。

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。

创建修订版时，可以修改以下容器属性和环境属性。

- 容器镜像 URI
- 端口映射
- 环境变量
- 任务大小
- 容器大小

- 任务角色
- 任务执行角色
- 卷和容器挂载点
- 私有注册表

JSON 验证

Amazon ECS 控制台 JSON 编辑器会在 JSON 文件中验证以下各项：

- 该文件是有效的 JSON 文件
- 该文件不包含任何无关的键
- 该文件包含 `familyName` 参数
- `containerDefinitions` 下方至少有一个条目

过程

Amazon ECS console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择包含您的任务定义的地区。
3. 在导航窗格中，选择 Task definitions（任务定义）。
4. 选择任务定义。
5. 选择任务定义修订版，然后选择创建新的修订、创建新的修订。
6. 在 Create new task definition revision（创建新任务定义修订版）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，然后进行更改。
7. 验证信息，然后选择更新。
8. 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 使用控制台更新 Amazon ECS 服务。

Amazon ECS console JSON editor

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。

3. 选择 Create new revision（创建新的修订）、Create new revision with JSON（使用 JSON 创建新的修订）。
4. 在 JSON 编辑器框中，编辑您的 JSON 文件，

   JSON 必须通过 the section called “JSON 验证” 中指定的验证检查。
5. 选择创建。

使用控制台注销 Amazon ECS 任务定义修订

当 Amazon ECS 中不再需要特定任务定义修订时，您可以注销任务定义修订，这样一来，当您想运行任务或更新服务时，任务定义修订将不再显示在 `ListTaskDefinition` API 调用或控制台中。

在注销任务定义修订后，它将立即被标记为 `INACTIVE`。现有任务和服务引用 `INACTIVE` 任务定义修订版将继续运行，而不会中断。引用 `INACTIVE` 任务定义修订仍可通过修改服务的预期数目来向上扩展或向下扩展。

您不能使用 `INACTIVE` 任务定义修订版以运行新任务或创建新服务。您也无法更新现有服务以引用 `INACTIVE` 任务定义修订版（即使在取消注册后，这些限制尚未生效的情况下，可能会有长达10分钟的窗口）。

> **Note**
>
> 注销任务系列中的所有修订后，任务定义系列将移至 `INACTIVE` 列表。添加 `INACTIVE` 任务定义的新修订会将任务定义系列移回 `ACTIVE` 列表。
> 在这个时候,`INACTIVE` 任务定义修订版本仍然可以无限期地在您的账户中发现。但是，此行为可能会在将来发生变化。因此，您不应该依赖 `INACTIVE` 任务定义修订版，这些修订版将持续存在于任何相关任务和服务的生命周期之外。

AWS CloudFormation 堆栈

以下行为适用于 2023 年 1 月 12 日之前在新的 Amazon ECS 控制台中创建的任务定义。

当您创建任务定义时，Amazon ECS 控制台会自动创建一个名称以 `ECS-Console-V2-TaskDefinition-` 开头的 CloudFormation 堆栈。如果您使用 AWS CLI 或 AWS SDK 取消注册任务定义，则必须手动删除任务定义堆栈。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的删除堆栈。

2023 年 1 月 12 日之后创建的任务定义不会为其自动创建 CloudFormation 堆栈。

过程

取消注册新的任务定义（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
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（任务定义）。
4. 在 Task definitions（任务定义）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系列。
5. 在任务定义名称页面上，选择要删除的修订，然后依次选择操作、取消注册。
6. 验证 Deregister（注销）窗口中的信息，然后选择 Deregister（注销）以完成操作。

使用控制台删除 Amazon ECS 任务定义修订

当 Amazon ECS 中不再需要特定的任务定义修订时，您可以删除该任务定义修订。

删除任务定义修订时，它将立即从 `INACTIVE` 转换为 `DELETE_IN_PROGRESS`。引用 `DELETE_IN_PROGRESS` 任务定义修订的现有任务和服务将继续运行，而不会中断。

您不能使用 `DELETE_IN_PROGRESS` 任务定义修订来运行新任务或创建新服务。您也无法更新现有服务以引用 `DELETE_IN_PROGRESS` 任务定义修订。

删除所有 `INACTIVE` 任务定义修订版时，任务定义名称不会显示在控制台中，也不会在 API 中返回。如果任务定义修订版处于 `DELETE_IN_PROGRESS` 状态，任务定义名称会显示在控制台中，并且在 API 中返回。任务定义名称由 Amazon ECS 保留，并且在下次使用该名称创建任务定义时，修订版本会增加。

可以阻止删除的 Amazon ECS 资源

当有任何依赖于任务定义修订的 Amazon ECS 资源时，将无法完成任务定义删除请求。以下资源可能会阻止任务定义被删除：

- Amazon ECS 任务 - 需要任务定义才能使任务保持正常运行。
- Amazon ECS 部署和任务集 - 在 Amazon ECS 部署或任务集启动扩展事件时，需要任务定义。

如果您的任务定义仍处于 `DELETE_IN_PROGRESS` 状态，则可以使用控制台或 AWS CLI 来识别，然后停止阻止删除任务定义的资源。

移除被阻止的资源后删除任务定义

在您移除阻止删除任务定义的资源后，将适用以下规则：

- Amazon ECS 任务 - 任务停止后，删除任务定义最多可能需要 1 小时才能完成。
- Amazon ECS 部署和任务集 - 删除部署或任务集后，任务定义删除最多可能需要 24 小时才能完成。

过程

删除任务定义（Amazon ECS 控制台）

在删除任务定义修订之前，您必须将其取消注册。有关更多信息，请参阅 the section called “使用控制台取消注册任务定义修订”。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
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（任务定义）。
4. 在任务定义页面上，选择包含要删除的一个或多个修订的任务定义系列。
5. 在任务定义名称页面上，选择要删除的修订，然后依次选择操作、删除。

   如果删除不可用，您必须注销任务定义。
6. 验证删除确认框中的信息，然后选择删除以完成操作。

Amazon ECS 任务定义应用场景

了解如何为各种 AWS 服务和功能编写任务定义的更多信息。

根据您的工作负载，需要设置某些任务定义参数。另外，对于 EC2 启动类型，您必须选择专为工作负载设计的特定实例。

主题

- 适用于 GPU 工作负载的 Amazon ECS 任务定义
- 适用于视频转码工作负载的 Amazon ECS 任务定义
- 适用于 AWS 神经元机器学习工作负载的 Amazon ECS 任务定义
- 适用于深度学习实例的 Amazon ECS 任务定义
- 适用于 64 位 ARM 工作负载的 Amazon ECS 任务定义

- 将 Amazon ECS 日志发送到 CloudWatch
- 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner
- 在 Amazon ECS 中使用非 AWS 容器映像
- 将单个环境变量传递给 Amazon ECS 容器
- 将环境变量传递给 Amazon ECS 容器
- 将敏感数据传递给 Amazon ECS 容器

适用于 GPU 工作负载的 Amazon ECS 任务定义

当您使用支持 GPU 的容器实例创建集群时，Amazon ECS 支持使用 GPU 的工作负载。使用 p2、p3、p5、g3、g4 和 g5 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》中的 Linux 加速型计算实例。

Amazon ECS 提供了经 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 经 Amazon ECS 优化的 Linux AMI。

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 将安排支持 GPU 的可用容器实例，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、Amazon EC2 P4d 实例、Amazon EC2 P5 实例、Amazon EC2 G3 实例、Amazon EC2 G4 实例、Amazon EC2 G5 实例和 Amazon EC2 G6 实例。

实例类型	GPU	GPU 内存 (GiB)	vCPU	内存 (GiB)
p3.2xlarge	1	16	8	61
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048
g3s.xlarge	1	8	4	30.5

实例类型	GPU	GPU 内存 (GiB)	vCPU	内存 (GiB)
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488
g4dn.xlarge	1	16	4	16
g4dn.2xlarge	1	16	8	32
g4dn.4xlarge	1	16	16	64
g4dn.8xlarge	1	16	32	128
g4dn.12xlarge	4	64	48	192
g4dn.16xlarge	1	16	64	256
g5.xlarge	1	24	4	16
g5.2xlarge	1	24	8	32
g5.4xlarge	1	24	16	64
g5.8xlarge	1	24	32	128
g5.16xlarge	1	24	64	256
g5.12xlarge	4	96	48	192
g5.24xlarge	4	96	96	384
g5.48xlarge	8	192	192	768
g6.xlarge	1	24	4	16
g6.2xlarge	1	24	8	32
g6.4xlarge	1	24	16	64

实例类型	GPU	GPU 内存 (GiB)	vCPU	内存 (GiB)
g6.8xlarge	1	24	32	128
g6.16.xlarge	1	24	64	256
g6.12xlarge	4	96	48	192
g6.24xlarge	4	96	48	192
g6.48xlarge	8	192	192	768
g6.metal	8	192	192	768
gr6.4xlarge	1	24	16	128
gr6.8xlarge	1	24	32	256

您可以通过查询 AWS Systems Manager Parameter Store API 来检索经 Amazon EKS 优化的 AMI 的亚马逊机器映像（AMI）ID。使用此参数，您无需手动查找经 Amazon ECS 优化的 AMI ID。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameter。您使用的用户必须具有 `ssm:GetParameter` IAM 权限才能检索经 Amazon ECS 优化的 AMI 元数据。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

注意事项

> **Note**
>
> 已弃用对 g2 实例系列类型的支持。
>
> Amazon ECS 经 GPU 优化的 AMI 的 20230912 版本之前的版本仅支持 p2 实例系列类型。如果您需要继续使用 p2 实例，请参阅 如果您需要 P2 实例该怎么办。
>
> 这两种实例系列类型上的 NVIDIA/CUDA 驱动程序的就地更新将导致 GPU 工作负载出现潜在故障。

开始在 Amazon ECS 上使用 GPU 之前，我们建议您考虑以下事项。

- 您的集群可以包含 GPU 和非 GPU 容器实例的组合。
- 您可以在外部实例上运行 GPU 工作负载。当向集群注册外部实例时，请确保安装脚本中包含 `--enable-gpu` 标记。有关更多信息，请参阅 将外部实例注册到 Amazon ECS 集群。
- 您必须在代理配置文件中将 `ECS_ENABLE_GPU_SUPPORT` 设置为 `true`。有关更多信息，请参阅 the section called “容器代理配置”。
- 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，以确定要在其上启动任务的容器实例。通过这样做，您可以更有效地使用您的资源。有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

  以下示例在您的默认集群中的 `g4dn.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 ==
    g4dn.xlarge" --region us-east-2
  ```

- 对于在容器定义中指定了 GPU 资源要求的每个容器，Amazon ECS 将容器运行时设置为 NVIDIA 容器运行时。
- NVIDIA 容器运行时需要在容器中设置一些环境变量才能工作。有关这些环境变量的列表，请参阅使用 Docker 的专用配置。Amazon ECS 设置 `NVIDIA_VISIBLE_DEVICES` 环境变量值设置为 Amazon ECS 分配给容器的 GPU 设备 ID 列表。对于其他必需的环境变量，Amazon ECS 不会对其进行设置。因此，请确保容器映像对其进行设置，或者在容器定义中对其进行设置。
- `20230929` 版本和更高版本的 Amazon ECS 经 GPU 优化的 AMI 支持 p5 实例类型系列。
- `20230913` 版本和更高版本的 Amazon ECS 经 GPU 优化的 AMI 支持 g4 实例类型系列。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。它在 Amazon ECS 控制台的“创建集群”工作流中不受支持。要使用这些实例类型，您必须使用 Amazon EC2 控制台、AWS CLI 或 API 并手动将实例注册到您的集群。
- p4d.24xlarge 实例类型仅适用于 CUDA 11 或更高版本。
- Amazon ECS 经 GPU 优化的 AMI 启用了 IPv6，这会导致使用 `yum` 时出现问题。可以通过配置 `yum` 将 IPv4 与以下命令结合使用。

  ```
  echo "ip_resolve=4" >> /etc/yum.conf
  ```

- 当您构建不使用 NVIDIA/CUDA 基础映像的容器映像时，必须设置 `NVIDIA_DRIVER_CAPABILITIES` 容器运行时变量设置为以下值之一：
  - `utility,compute`
  - `all`

有关如何设置变量的信息，请参阅 NVIDIA 网站上的控制 NVIDIA 容器运行时

- Windows 容器不支持 GPU。

启动适用于 Amazon ECS 的 GPU 容器实例

要在 Amazon ECS 上使用 GPU 实例，您需要创建启动模板、用户数据文件并启动该实例。

然后，您可以运行使用为 GPU 配置的任务定义的任务。

使用启动模板

您可以创建启动模板。

- 创建将经 Amazon ECS 优化的 GPU AMI ID 用于 AMI 的启动模板。有关如何创建启动模板的信息，请参阅《Amazon EC2 用户指南》中的使用您定义的参数创建新启动模板。

  将上一步中的 AMI ID 用于亚马逊机器映像。有关如何使用 Systems Manager 参数指定 AMI ID 的信息，请参阅《Amazon EC2 用户指南》中的在启动模板中指定 Systems Manager 参数。

  将以下各项添加到启动模板中的用户数据中。将 *cluster-name* 替换为您集群的名称。

  ```
  #!/bin/bash
  echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
  echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
  ```

使用 AWS CLI

您可以使用 AWS CLI 启动容器实例。

1. 创建名为 `userdata.toml` 的文件。此文件会用于实例用户数据。将 *cluster-name* 替换为您集群的名称。

   ```
   #!/bin/bash
   echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
   echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
   ```

2. 运行以下命令以获取 GPU AMI ID。您将在以下步骤中使用此 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

3. 运行以下命令来启动 GPU 实例。请记得替换以下参数：

   - 用您的实例将在其中启动的私有或公有子网的 ID 替换*##*。
   - 将 *gpu_ami* 替换为上一步中的 AMI ID。
   - 将 *t3.large* 替换为您要使用的实例类型。
   - 将 *region* 替换为区域代码。

```
aws ec2 run-instances --key-name ecs-gpu-example \
   --subnet-id subnet \
   --image-id gpu_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications 'ResourceType=instance,Tags=[{Key=GPU,Value=example}]' \
   --user-data file://userdata.toml \
   --iam-instance-profile Name=ecsInstanceRole
```

4. 运行以下命令，以验证容器实例是否注册到集群。在运行此命令时，请记得替换以下参数：

   - 将 *cluster* 替换为您的集群名称。
   - 将 *region* 替换为区域代码。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

在 Amazon ECS 任务定义中指定 GPU 数

要在容器实例和 Docker GPU 运行时上使用 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，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 的数量不能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 使用控制台创建 Amazon ECS 任务定义。

> **Important**
>
> 如果任务定义中未指定 GPU 要求，则任务将使用原定设置 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。此容器使用两个 GPU，运行 `nvidia-smi` 实用程序，然后退出。

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:11.0.3-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
  ],
  "family": "example-ecs-gpu"
}
```

如果您需要 P2 实例该怎么办

如果您需要使用 P2 实例，可以使用以下选项之一继续使用实例。

您必须修改这两个选项的实例用户数据。有关更多信息，请参阅《Amazon EC2 用户指南》中的使用实例用户数据。

使用最新支持的经 GPU 优化的 AMI

您可以使用经 GPU 优化的 AMI 的 20230906 版本，并将以下内容添加到实例用户数据中。

将 cluster-name 替换为您集群的名称。

```
#!/bin/bash
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
```

使用最新的经 GPU 优化的 AMI，并更新用户数据

您可在实例用户数据中添加以下内容。这将卸载 Nvidia 535/Cuda12.2 驱动程序，然后安装 Nvidia 470/Cuda11.4 驱动程序并修复该版本。

```
#!/bin/bash
yum remove -y cuda-toolkit* nvidia-driver-latest-dkms*
tmpfile=$(mktemp)
cat >$tmpfile <<EOF
[amzn2-nvidia]
name=Amazon Linux 2 Nvidia repository
mirrorlist=\$awsproto://\$amazonlinux.\$awsregion.\$awsdomain/\$releasever/amzn2-
nvidia/latest/\$basearch/mirror.list
priority=20
gpgcheck=1
gpgkey=https://developer.download.nvidia.com/compute/cuda/repos/rhel7/
x86_64/7fa2af80.pub
enabled=1
exclude=libglvnd-*
EOF

mv $tmpfile /etc/yum.repos.d/amzn2-nvidia-tmp.repo
yum install -y system-release-nvidia cuda-toolkit-11-4 nvidia-driver-latest-
dkms-470.182.03
yum install -y libnvidia-container-1.4.0 libnvidia-container-tools-1.4.0 nvidia-
container-runtime-hook-1.4.0 docker-runtime-nvidia-1
```

```
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
nvidia-smi
```

创建您自己的兼容 P2 且经过 GPU 优化的 AMI

您可以创建与 P2 实例兼容的自定义 Amazon ECS GPU 优化型 AMI，然后使用 AMI 启动 P2 实例。

1. 运行以下命令以克隆 `amazon-ecs-ami repo`。

   ```
   git clone https://github.com/aws/amazon-ecs-ami
   ```

2. 在 `release.auto.pkrvars.hcl` 或 `overrides.auto.pkrvars.hcl` 中设置所需的 Amazon ECS 代理和源 Amazon Linux AMI 版本。
3. 运行以下命令来构建兼容 P2 的私有 EC2 AMI。

   将“区域”替换为“区域”和“实例区域”。

   ```
   REGION=region make al2keplergpu
   ```

4. 使用包含以下实例用户数据的 AMI 连接到 Amazon ECS 集群。

   将 cluster-name 替换为您集群的名称。

   ```
   #!/bin/bash
   echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
   ```

适用于视频转码工作负载的 Amazon ECS 任务定义

要在 Amazon ECS 上使用视频转码工作负载，请注册 Amazon EC2 VT1 实例。注册这些实例后，您可以将实时和预渲染的视频转码工作负载作为 Amazon ECS 上的任务运行。Amazon EC2 VT1 实例使用 Xilinx U30 媒体转码卡来加速实时和预渲染的视频转码工作负载。

> **Note**
>
> 有关如何在 Amazon ECS 以外的容器中运行视频转码工作负载的说明，请参阅 Xilinx 文档。

注意事项

在 Amazon ECS 上开始部署 VT1 之前，请注意以下事项：

- 您的集群可以包含 VT1 和非 VT1 实例的组合。
- 您需要一个 Linux 应用程序，该应用程序使用具有加速 AVC（H.264）和 HEVC（H.265）编解码器的 Xilinx U30 媒体转码卡。

> **Important**
>
> 使用其他编解码器的应用程序可能不会在 VT1 实例上获得性能提升。

- U30 卡上只能运行一个转码任务。每张卡都有两个与之关联的设备。您可以运行的转码任务数量与每个 VT1 实例的卡的数量一样。
- 在创建服务或运行独立任务时，您可以在配置任务放置约束时使用实例类型属性。这样可以确保在您指定的容器实例上启动任务。这样做有助于确保您有效地使用资源，并确保视频转码工作负载任务在 VT1 实例上。有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

  在以下示例中，任务在您的 `default` 集群上的 `vt1.3xlarge` 实例上运行。

```
aws ecs run-task \
     --cluster default \
     --task-definition vt1-3xlarge-xffmpeg-processor \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 vt1.3xlarge"
```

- 您可以将容器配置为使用主机容器实例上可用的特定 U30 卡。您可以通过使用 `linuxParameters` 参数并指定设备详细信息来执行此操作。有关更多信息，请参阅 任务定义要求。

使用 VT1 AMI

您有两种选择可以在 Amazon EC2 上运行适用于 Amazon ECS 容器实例的 AMI。第一个选项是在 AWS Marketplace 上使用 Xilinx 官方 AMI。第二种选择是从示例存储库构建自己的 AMI。

- Xilinx 在 AWS Marketplace 上提供 AMI。
- Amazon ECS 提供了一个示例存储库，您可以使用该存储库为视频转码工作负载构建 AMI。该 AMI 随附 Xilinx U30 驱动程序。您可以在 Github 上找到包含 Packer 脚本的存储库。有关 Packer 的更多信息，请参阅 Packer 文档。

任务定义要求

要在 Amazon ECS 上运行视频转码容器，您的任务定义必须包含使用加速 H.264/AVC 和 H.265/HEVC 编解码器的视频转码应用程序。您可以按照 Xilinx Github 上的步骤来构建容器映像。

任务定义必须特定于实例类型。实例类型为 3xlarge、6xlarge 和 24xlarge。您必须将容器配置为使用主机容器实例上可用的特定 Xilinx U30 设备。您还可以使用 `linuxParameters` 参数执行此操作。下表详细介绍了特定于每种实例类型的卡和设备 SoC。

实例类型	vCPU	RAM (GiB)	U30 加速器卡	可寻址的 XCU30 SoC 设备	设备路径
vt1.3xlarge	12	24	1	2	`/dev/dri/renderD128 ,/dev/dri/renderD129`
vt1.6xlarge	24	48	2	4	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131`
vt1.24xlarge	96	182	8	16	`/dev/dri/renderD128 ,/dev/dri/renderD12`

实例类型	vCPU	RAM (GiB)	U30 加速器卡	可寻址的 XCU30 SoC 设备	设备路径
					9 ,/dev/dri/renderD130 ,/dev/dri/renderD131 ,/dev/dri/renderD132 ,/dev/dri/renderD133 ,/dev/dri/renderD134 ,/dev/dri/renderD135 ,/dev/dri/renderD136 ,/dev/dri/renderD137 ,/dev/dri/renderD138 ,/dev/dri/renderD139 ,/dev/dri/

实例类型	vCPU	RAM (GiB)	U30 加速器卡	可寻址的 XCU30 SoC 设备	设备路径
					renderD140 ,/dev/dri/renderD141 ,/dev/dri/renderD142 ,/dev/dri/renderD143

Important

如果任务定义列出了 EC2 实例没有的设备，则任务将无法运行。当任务失败时，stoppedReason 中将出现以下错误消息：CannotStartContainerError: Error response from daemon: error gathering device information while adding custom device "/dev/dri/renderD*130*": no such file or directory。

在 Amazon ECS 任务定义中指定视频转码

以下示例中，提供了用于 Amazon EC2 上 Linux 容器的任务定义的语法。此任务定义适用于按照 Xilinx 文档中提供的过程构建的容器映像。如果使用此示例，请将 image 替换为您自己的映像，然后将视频文件复制到 /home/ec2-user 目录中的实例。

vt1.3xlarge

1. 使用以下内容创建名为 vt1-3xlarge-ffmpeg-linux.json 的文本文件。

```
{
    "family": "vt1-3xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
```

```
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.3xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
```

```
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 注册任务定义。

```
aws ecs register-task-definition --family vt1-3xlarge-xffmpeg-processor --cli-
input-json file://vt1-3xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.6xlarge

1. 使用以下内容创建名为 vt1-6xlarge-ffmpeg-linux.json 的文本文件。

```
{
    "family": "vt1-6xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.6xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
```

```
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
```

```
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 注册任务定义。

```
aws ecs register-task-definition --family vt1-6xlarge-xffmpeg-processor --cli-
input-json file://vt1-6xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.24xlarge

1. 使用以下内容创建名为 vt1-24xlarge-ffmpeg-linux.json 的文本文件。

```
{
    "family": "vt1-24xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.24xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
```

```
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD132",
                        "hostPath": "/dev/dri/renderD132",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
```

```
                    {
                        "containerPath": "/dev/dri/renderD133",
                        "hostPath": "/dev/dri/renderD133",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD134",
                        "hostPath": "/dev/dri/renderD134",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD135",
                        "hostPath": "/dev/dri/renderD135",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD136",
                        "hostPath": "/dev/dri/renderD136",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD137",
                        "hostPath": "/dev/dri/renderD137",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD138",
                        "hostPath": "/dev/dri/renderD138",
                        "permissions": [
```

```
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD139",
                    "hostPath": "/dev/dri/renderD139",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD140",
                    "hostPath": "/dev/dri/renderD140",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD141",
                    "hostPath": "/dev/dri/renderD141",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD142",
                    "hostPath": "/dev/dri/renderD142",
                    "permissions": [
                        "read",
                        "write"
                    ]
                },
                {
                    "containerPath": "/dev/dri/renderD143",
                    "hostPath": "/dev/dri/renderD143",
                    "permissions": [
                        "read",
                        "write"
                    ]
                }
```

```
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 注册任务定义。

```
aws ecs register-task-definition --family vt1-24xlarge-xffmpeg-processor --cli-
input-json file://vt1-24xlarge-xffmpeg-linux.json --region us-east-1
```

适用于 AWS 神经元机器学习工作负载的 Amazon ECS 任务定义

您可以将 Amazon EC2 Trn1、Amazon EC2 Inf1 和 Amazon EC2 Inf2 实例注册到您的集群，以执行机器学习工作负载。

Amazon EC2 Trn1 实例由 AWS Trainium 芯片提供技术支持。这些实例在云中为机器学习提供高性能的低成本训练。您可以在 Trn1 实例上使用具有 AWS Neuron 的机器学习框架训练机器学习推理模型。然后，您可以在 Inf1 实例或 Inf2 实例上运行模型来利用 AWS Inferentia 芯片的加速。

Amazon EC2 Inf1 实例和 Inf2 实例由 AWS Inferentia 芯片提供支持，，它们可在云中提供高性能和最低成本的推理。

机器学习模型使用 AWS Neuron 部署到容器中，它是专门的软件开发工具包（SDK）。SDL 由编译器、运行时和分析工具组成，可用于优化 AWS 机器学习芯片的机器学习性能。AWSNeuron 支持常用的机器学习框架，例如 TensorFlow、PyTorch 和 Apache MXNet。

注意事项

在 Amazon ECS 上开始部署 Neuron 之前，请注意以下事项：

- 您的集群可以包含 Trn1、Inf1、Inf2 和其他实例的组合。
- 在容器中，您需要一个采用支持 AWS Neuron 的机器学习框架的 Linux 应用程序。

> **Important**
>
> 使用其他框架的应用程序可能不会在 Trn1、Inf1 和 Inf2 实例上获得性能提升。

- 每个 AWS Trainium 或 AWS Inferentia 芯片上只能运行一个推理或推理训练任务。对于 Inf1，每个芯片有 4 个 NeuronCore。对于 Trn1 和 Inf2，每个芯片有 2 个 NeuronCore。每个 Trn1、Inf1 和 Inf2 实例都有芯片，您可以运行与之一样多的任务。
- 在创建服务或运行独立任务时，您可以在配置任务放置约束时使用实例类型属性。这样可以确保在您指定的容器实例上启动任务。这样做可以帮助您优化总体资源利用率，并确保推理工作负载的任务位于 Trn1、Inf1 和 Inf2 实例上。有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

  在以下示例中，任务在您的 `default` 集群上的 `Inf1.xlarge` 实例上运行。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-inference-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 Inf1.xlarge"
```

- 无法在任务定义中定义 Neuron 资源需求。但您要将容器配置为使用主机容器实例上可用的特定 AWS Trainium 或 AWSInferentia 芯片。通过使用 `linuxParameters` 参数并指定设备详细信息来执行此操作。有关更多信息，请参阅 任务定义要求。

使用经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI

Amazon ECS 为 AWS Trainium 和 AWS Inferentia 工作负载提供了一个基于 Amazon Linux 2023 的经 Amazon ECS 优化的 AMI。它附带适用于 Docker 的 AWS Neuron 驱动程序和运行时。此 AMI 使得在 Amazon ECS 上运行机器学习 inference 工作负载变得更加轻松。

我们建议您在启动 Amazon EC2 Trn1、Inf1 和 Inf2 实例时使用经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI。

您可以使用 AWS CLI 和以下命令检索当前经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/recommended
```

以下区域支持经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（东京)
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（巴黎）
- 欧洲（斯德哥尔摩）
- 南美洲（圣保罗）

使用经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI

Amazon ECS 为 AWS Trainium 和 AWS Inferentia 工作负载提供了一个基于 Amazon Linux 2 的经 Amazon ECS 优化的 AMI。它附带适用于 Docker 的 AWS Neuron 驱动程序和运行时。此 AMI 使得在 Amazon ECS 上运行机器学习 inference 工作负载变得更加轻松。

建议您在启动 Amazon EC2 Trn1、Inf1 和 Inf2 实例时使用经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI。

您可以使用 AWS CLI 和以下命令检索当前经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended
```

以下区域支持经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（东京)
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 加拿大（中部）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（巴黎）
- 欧洲（斯德哥尔摩）
- 南美洲（圣保罗）

任务定义要求

要在 Amazon ECS 上部署 Neuron，您的任务定义必须包含预构建容器的容器定义，该容器服务于 TensorFlow 的 推理模型。它是由 AWS 深度学习容器提供的。此容器包含 AWS Neuron 运行时和 TensorFlow 服务应用程序。在启动时，此容器将从 Amazon S3 获取您的模型，用保存的模型启动 Neuron TensorFlow 服务，并等待预测请求。在以下示例中，容器映像具有 TensorFlow 1.15 和 Ubuntu 18.04。GitHub 上维护了针对 Neuron 优化的预构建 Deep Learning Containers 的完整列表。有关更多信息，请参阅使用 AWS Neuron TensorFlow Serving。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-neuron:1.15.4-neuron-py37-ubuntu18.04
```

或者，您也可以构建自己的 Neuron 边车容器映像。有关更多信息，请参阅《AWS Deep Learning AMI 开发人员指南》中的教程：Neuron TensorFlow Serving。

任务定义必须特定于一种实例类型。您必须将容器配置为使用主机容器实例上可用的特定 AWS Trainium 或 AWS Inferentia 设备。您还可以使用 `linuxParameters` 参数执行此操作。下表详细介绍了特定于每种实例类型的芯片。

实例类型	vCPU	RAM (GiB)	AWS ML 加速器芯片	设备路径
trn1.2xlarge	8	32	1	`/dev/neuron0`
trn1.32xlarge	128	512	16	`/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neur`

实例类型	vCPU	RAM (GiB)	AWS ML 加速器芯片	设备路径
				on9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf1.xlarge	4	8	1	/dev/neuron0
inf1.2xlarge	8	16	1	/dev/neuron0
inf1.6xlarge	24	48	4	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3

实例类型	vCPU	RAM (GiB)	AWS ML 加速器芯片	设备路径
inf1.24xlarge	96	192	16	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf2.xlarge	8	16	1	/dev/neuron0
inf2.8xlarge	32	64	1	/dev/neuron0

实例类型	vCPU	RAM (GiB)	AWS ML 加速器芯片	设备路径
inf2.24xlarge	96	384	6	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,
inf2.48xlarge	192	768	12	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11

在 Amazon ECS 任务定义中指定 AWS Neuron 机器学习

以下是 `inf1.xlarge` 的示例 Linux 任务定义，显示要使用的语法。

```
{
    "family": "ecs-neuron",
    "requiresCompatibilities": ["EC2"],
```

```
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == inf1.xlarge"
        }
    ],
    "executionRoleArn": "${YOUR_EXECUTION_ROLE}",
    "containerDefinitions": [
        {
            "entryPoint": [
                "/usr/local/bin/entrypoint.sh",
                "--port=8500",
                "--rest_api_port=9000",
                "--model_name=resnet50_neuron",
                "--model_base_path=s3://your-bucket-of-models/resnet50_neuron/"
            ],
            "portMappings": [
                {
                    "hostPort": 8500,
                    "protocol": "tcp",
                    "containerPort": 8500
                },
                {
                    "hostPort": 8501,
                    "protocol": "tcp",
                    "containerPort": 8501
                },
                {
                    "hostPort": 0,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/neuron0",
                        "hostPath": "/dev/neuron0",
                        "permissions": [
                            "read",
```

```
                            "write"
                        ]
                    }
                ],
                "capabilities": {
                    "add": [
                        "IPC_LOCK"
                    ]
                }
            },
            "cpu": 0,
            "memoryReservation": 1000,
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference-neuron:1.15.4-neuron-py37-ubuntu18.04",
            "essential": true,
            "name": "resnet50"
        }
    ]
}
```

适用于深度学习实例的 Amazon ECS 任务定义

要在 Amazon ECS 上使用深度学习工作负载，请将 Amazon EC2 DL1 实例注册到您的集群。Amazon EC2 DL1 实例由英特尔旗下公司 Habana Labs 的 Gaudi 加速器提供支持。使用 Habana SynapseAI SDK 连接 Habana Gaudi 加速器。SDK 支持常用的机器学习框架 TensorFlow 和 PyTorch。

注意事项

在 Amazon ECS 上开始部署 DL1 之前，请注意以下事项：

- 您的集群可以包含 DL1 和非 DL1 实例的组合。
- 在创建服务或运行独立任务时，您可以在配置任务放置约束时使用实例类型属性，以确保在您指定的容器实例上启动任务。这样做可以确保您有效地使用资源，并确保深度学习工作负载任务在 DL1 实例上。有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

  以下示例运行 `default` 集群上的 `dl1.24xlarge` 实例任务。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-dl1-task-def \
```

```
    --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
dl1.24xlarge"
```

使用 DL1 AMI

您有三种选择可以在 Amazon EC2 DL1 实例上运行适用于 Amazon ECS 的 AMI：

- Habana 在此处提供的 AWS Marketplace AMI。
- Amazon Web Services 提供的 Habana 深度学习 AMI。因为它不包括在内，因此您需要单独安装 Amazon ECS 容器代理。
- 使用 Packer 构建由 GitHub 存储库提供的自定义 AMI。有关更多信息，请参阅 Packer 文档。

在 Amazon ECS 任务定义中指定深度学习

要在 Amazon ECS 上运行 Habana Gaudi 加速深度学习容器，您的任务定义必须包含预构建容器的容器定义，该容器使用 AWS 深度学习容器提供的 Habana SynapseAI 服务于 TensorFlow 或 PyTorch 的深度学习模型。

以下容器映像具有 TensorFlow 2.7.0 和 Ubuntu 20.04。GitHub 上维护了针对 Habana Gaudi 加速器优化的预构建深度学习容器的完整列表。有关更多信息，请参阅 Habana Training Containers（Habana 训练容器）。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training-habana:2.7.0-hpu-py38-
synapseai1.2.0-ubuntu20.04
```

以下是 Amazon EC2 上的 Linux 容器的示例任务定义，显示要使用的语法。此示例使用包含 Habana Labs 实验室系统管理界面工具（HL-SMI）的图像，可参见此处：`vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/tensorflow-installer-tf-cpu-2.6.0:1.1.0-614`

```
{
    "family": "dl-test",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
```

```
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == dl1.24xlarge"
        }
    ],
    "networkMode": "host",
    "cpu": "10240",
    "memory": "1024",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": ["hl-smi"],
            "cpu": 8192,
            "environment": [
                {
                    "name": "HABANA_VISIBLE_DEVICES",
                    "value": "all"
                }
            ],
            "image": "vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/
tensorflow-installer-tf-cpu-2.6.0:1.1.0-614",
            "essential": true,
            "name": "tensorflow-installer-tf-hpu"
        }
    ]
}
```

适用于 64 位 ARM 工作负载的 Amazon ECS 任务定义

Amazon ECS 支持使用 64 位 ARM 应用程序。您可以在 AWS Graviton2 处理器支持的平台上运行您的应用程序。它适用于各类工作负载。这包括应用程序服务器、微服务、高性能计算、基于 CPU 的机器学习推断、视频编码、电子设计自动化、游戏、开源数据库和内存缓存等工作负载。

注意事项

在开始部署使用 64 位 ARM 架构的任务定义之前，请考虑以下事项：

- 应用程序可以使用 Fargate 或 EC2 启动类型。
- 使用 ARM64 架构的 Linux 任务不支持 Fargate Spot 容量提供程序。

- 这些应用程序只能使用 Linux 操作系统。
- 对于 Fargate 类型，应用程序必须使用 Fargate 平台版本 `1.4.0` 或更高版本。
- 应用程序可以使用 Fluent Bit 或 CloudWatch 进行监控。
- 对于 Fargate 启动类型，以下AWS 区域不支持 64 位 ARM 工作负载：
  - 美国东部（弗吉尼亚州北部），`use1-az3` 可用区
- 对于 Amazon EC2 启动类型，请参阅以下内容以验证您的区域是否支持要使用的实例类型：
  - Amazon EC2 M6g 实例
  - Amazon EC2 T4g 实例
  - Amazon EC2 C6g 实例
  - Amazon EC2 R6gd 实例
  - Amazon EC2 X2gd 实例

  您还可以使用带筛选器的 Amazon EC2 `describe-instance-type-offerings` 命令来查看您所在区域的实例产品。

  ```
  aws ec2 describe-instance-type-offerings --filters Name=instance-
  type,Values=instance-type --region region
  ```

  以下示例检查美国东部（弗吉尼亚州北部）（us-east-1）区域中的 M6 实例类型可用性。

  ```
  aws ec2 describe-instance-type-offerings --filters "Name=instance-type,Values=m6*" --
  region us-east-1
  ```

  有关更多信息，请参阅 Amazon EC2 命令行参考中的 describe-instance-type-offerings。

在 Amazon ECS 任务定义中指定 ARM 架构

要使用 ARM 架构，请为 `cpuArchitecture` 任务定义参数指定 `ARM64`。

在以下示例中，ARM 架构是在任务定义中指定的。该文件以 JSON 格式。

```
{
    "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
    },
```

```
...
}
```

在以下示例中，ARM 架构的任务定义显示“hello world”。

```
{
 "family": "arm64-testapp",
 "networkMode": "awsvpc",
 "containerDefinitions": [
    {
        "name": "arm-container",
        "image": "arm64v8/busybox",
        "cpu": 100,
        "memory": 100,
        "essential": true,
        "command": [ "echo hello world" ],
        "entryPoint": [ "sh", "-c" ]
    }
 ],
 "requiresCompatibilities": [ "FARGATE" ],
 "cpu": "256",
 "memory": "512",
 "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
  },
 "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

将 Amazon ECS 日志发送到 CloudWatch

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，则可以查看容器中的日志。如果您使用 EC2 启动类型，则可以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。

> **Note**
>
> 任务中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 Logs。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《Amazon CloudWatch Logs 用户指南》中的监控日志文件和 CloudWatch Logs 配额。

Fargate 启动类型

如果您对任务使用 Fargate 启动类型，您需要将所需的 `logConfiguration` 参数添加到您的任务定义才能打开 `awslogs` 日志驱动程序。有关更多信息，请参阅 Amazon ECS 任务定义示例：将日志路由到 CloudWatch。

对于 Fargate 上的 Windows 容器，当您的任何任务定义参数包含特殊字符（例如 `& \ < > ^ |`）时，请执行以下选项之一：

- 在整个参数字符串周围添加一个带有双引号的 escape（`\`）字符

  示例

  ```
  "awslogs-multiline-pattern": "\"^[|DEBUG|INFO|WARNING|ERROR\"",
  ```

- 在每个特殊字符周围添加一个 escape（`^`）字符

  示例

  ```
  "awslogs-multiline-pattern": "^^[^|DEBUG^|INFO^|WARNING^|ERROR",
  ```

EC2 启动类型

如果您对任务使用 EC2 启动类型，并且想要打开 `awslogs` 日志驱动程序，您的 Amazon ECS 容器实例至少需要容器代理的 1.9.0 版本。有关如何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。

> **Note**
>
> 您必须使用经 Amazon ECS 优化的 AMI 或至少包含 `ecs-init` 软件包版本 `1.9.0-1` 的自定义 AMI。使用自定义 AMI 时，如果在 docker run 语句或环境变量文件中使用以下环境变量启动代理，您必须指定 `awslogs` 日志驱动程序在 Amazon EC2 实例上可用。

```
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，可能需要添加此权限。`ecsTaskExecutionRole` 在将其分配给任务时使用，并且可能包含正确的权限。有关任务执行角色的信息，请参阅Amazon ECS 任务执行 IAM 角色。如果您的容器实例使用容器实例的托管 IAM policy，您的容器实例可能具有正确的权限。有关容器实例的托管 IAM 策略的信息，请参阅Amazon ECS 容器实例 IAM 角色。

Amazon ECS 任务定义示例：将日志路由到 CloudWatch

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。有关日志参数的更多信息，请参阅 存储和日志记录

下面的任务定义 JSON 有一个为每个容器指定的 `logConfiguration` 对象。一个是用于将日志发送到名为 `awslogs-wordpress` 的日志组的 WordPress 容器。另一个是用于将日志发送到名为 `awslogs-mysql` 的日志组的 MySQL 容器。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-create-group": "true",
                    "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-create-group": "true",
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example",
                    "mode": "non-blocking",
                    "max-buffer-size": "25m"
                }
            }
        }
    ],
    "family": "awslogs-example"
}
```

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，请参阅将应用程序作为 Amazon ECS 任务运行 和使用控制台创建 Amazon ECS 服务。

将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner

您可以使用适用于 Amazon ECS 的 FireLens 来使用任务定义参数将日志路由到 AWS 服务或 AWS Partner Network（APN）目标来进行日志存储和分析。AWS Partner Network 是一个由合作伙伴组成的全球社区，它利用计划、专业知识和资源来构建、营销和销售客户产品。有关更多信息，请参阅

AWS Partner。FireLens 与 Fluentd 和 Fluent Bit 结合使用。我们提供 AWS for Fluent Bit 映像，您也可以使用自己的 Fluentd 或 Fluent Bit 映像。

使用 FireLens for Amazon ECS 时考虑以下事项：

- 建议您在日志容器名称中添加 `my_service_`，以便可以在控制台中轻松区分容器名称。
- 默认情况下，Amazon ECS 在应用程序容器和 FireLens 容器之间添加启动容器顺序依赖关系。当您在应用程序容器和 FireLens 容器之间指定容器顺序时，默认的启动容器顺序将被覆盖。
- 支持将适用于 Amazon ECS 的 FireLens 用于托管在 Linux 上的 AWS Fargate 和 Amazon EC2 的任务。Windows 容器不支持 FireLens。

  有关如何为 Windows 容器配置集中日志记录的信息，请参阅使用 Fluent Bit 在 Amazon ECS 上对 Windows 容器进行集中日志记录。
- 您可以使用 AWS CloudFormation 模板为 Amazon ECS 配置 FireLens。有关更多信息，请参阅 AWS CloudFormation 用户指南中的 AWS::ECS::TaskDefinition FirelensConfiguration
- FireLens 在端口 24224 上侦听，因此为了确保从任务外部无法访问 FireLens 日志路由器，不得允许任务使用的安全组中端口 24224 上的入站流量。对于使用 `awsvpc` 网络模式的任务，这是与任务关联的安全组。对于使用 `host` 网络模式的任务，它是与托管任务的 Amazon EC2 实例关联的安全组。对于使用 `bridge` 网络模式的任务，请勿创建任何使用端口 24224 的端口映射。
- 对于使用 `bridge` 网络模式的任务，具有 FirelLens 配置的容器必须在依赖它的任何应用程序容器启动之前启动。要控制容器的启动顺序，请在任务定义中使用依赖条件。有关更多信息，请参阅 容器依赖项。

  > **Note**
  >
  > 如果您将容器定义中的依赖条件参数与 FirelLens 配置结合使用，请确保每个容器均具有 `START` 或 `HEALTHY` 条件要求。

- 默认情况下，FireLens 将集群和任务定义名称以及集群的 Amazon 资源名称（ARN）作为元数据键添加到您的 stdout/stderr 容器日志中。以下为元数据格式的示例。

```
"ecs_cluster": "cluster-name",
"ecs_task_arn": "arn:aws:ecs:region:111122223333:task/cluster-
name/f2ad7dba413f45ddb4EXAMPLE",
"ecs_task_definition": "task-def-name:revision",
```

  如果您不希望日志中出现元数据，请在任务定义的 `firelensConfiguration` 部分中将 `enable-ecs-log-metadata` 设置为 `false`。

```
"firelensConfiguration":{
   "type":"fluentbit",
   "options":{
      "enable-ecs-log-metadata":"false",
      "config-file-type":"file",
      "config-file-value":"/extra.conf"
}
```

要使用此功能，您必须为您的任务创建一个 IAM 角色，该角色提供使用任务需要的任何 AWS 服务所需的权限。例如，如果容器将日志路由到 Firehose，则任务需要调用 `firehose:PutRecordBatch` API 的权限。有关更多信息，请参阅 IAM用户指南中的添加和删除 IAM 标识权限。

在以下条件下，您的任务也可能需要 Amazon ECS 任务执行角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

- 如果在 Fargate 上托管您的任务，并且您正在从 Amazon ECR 提取容器映像或 AWS Secrets Manager 在日志配置中引用敏感数据，那么您必须包含任务执行 IAM 角色。
- 当您使用托管于 Amazon S3 中的自定义配置文件时，任务执行 IAM 角色必须包含 `s3:GetObject` 权限。

有关如何将多个配置文件用于 Amazon ECS 的信息（包括您托管的文件或 Amazon S3 中的文件），请参阅 Init process for Fluent Bit on ECS, multi-config support。

配置 Amazon ECS 日志，以实现高吞吐量

创建任务定义时，您可以通过在 `log-driver-buffer-limit` 中指定值来指定将在内存中缓冲的日志行数。有关更多信息，请参阅 Docker 文档中的 Fluentd 日志记录驱动程序。

建议在吞吐量高时使用此选项，因为 Docker 可能会耗尽缓冲区内存并丢弃缓冲区消息，以便添加新消息。

将 FireLens for Amazon ECS 与缓冲区限制选项结合使用时考虑以下事项：

- Amazon EC2 启动类型和带平台版本 `1.4.0` 或更高版本的 Fargate 启动类型支持此选项。
- 该选项仅在 `logDriver` 设置为 `awsfirelens` 时有效。
- 默认缓冲区限制为 `1048576` 个日志行。
- 有效值为 `0` 和 `536870912` 个日志行。

- 用于此缓冲区的最大内存量是每个日志行的大小与缓冲区大小的乘积。例如，假设应用程序的日志行大小平均为 2 KiB，则缓冲区限制为 4096 时最多只能使用 8 MiB。除了日志驱动程序的内存缓冲区外，在任务级别分配的内存总量必须大于为所有容器分配的内存量。

在任务定义中指定 `awsfirelens` 日志驱动程序时，Amazon ECS 容器 代理会将以下环境变量注入容器中：

`FLUENT_HOST`

分配给 FirelLens 容器的 IP 地址。

`FLUENT_PORT`

Fluent Forward 协议正在侦听的端口。

您可以使用 `FLUENT_HOST` 和 `FLUENT_PORT` 环境变量直接从代码登录到日志路由器，而不是通过 `stdout`。有关更多信息，请参阅 GitHub 上的 fluent-logger-golang。

下面说明指定 `log-driver-buffer-limit` 的语法。将 `my_service_` 替换为服务的名称：

```
{
    "containerDefinitions": [
        {
            "essential": true,
            "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
            "name": "my_service_log_router",
            "firelensConfiguration": {
                "type": "fluentbit"
            },
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "firelens-container",
                    "awslogs-region": "us-west-2",
                    "awslogs-create-group": "true",
                    "awslogs-stream-prefix": "firelens"
                }
            },
            "memoryReservation": 50
        },
        {
```

```
            "essential": true,
            "image": "httpd",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "51200"
                }
            },
            "dependsOn": [
                {
                    "containerName": "log_router",
                    "condition": "START"
                }
            ],
            "memoryReservation": 100
        }
    ]
}
```

Amazon ECS 的 AWS for Fluent Bit 映像存储库

AWS 提供了 Fluent Bit 映像以及 CloudWatch Logs 和 Firehose 的插件。我们建议使用 Fluent Bit 作为日志路由器，因为其资源利用率低于 Fluentd。有关更多信息，请参阅 CloudWatch Logs for Fluent Bit 和 Amazon Kinesis Firehose for Fluent Bit。

为了获得高可用性，大多数AWS 区域的 Amazon ECR 公共图库和 Amazon ECR 存储库上的 Amazon ECR 都提供了 AWS for Fluent Bit 映像。

Amazon ECR Public Gallery

Amazon ECR 公共图库提供了 AWS for Fluent Bit 映像。这是下载 AWS for Fluent Bit 映像的推荐位置，因为它是一个公共存储库，可从所有AWS 区域区域使用。有关更多信息，请参阅 Amazon ECR 公共图库上的 aws-for-fluent-bit。

Linux

Amazon ECR Public Gallery 中 Fluent Bit 映像的 AWS 支持带有 `ARM 64` 或 `x86-64` 架构的 Amazon Linux 操作系统。

通过使用所需的映像标记指定存储库 URL，可以从 Amazon ECR 公共图库中提取 AWS for Fluent Bit 映像。可在 Amazon ECR Public Gallery 上的映像标签选项卡上找到映像标签。

下面说明 Docker CLI 使用的语法。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

例如，您可以使用此 Docker CLI 命令拉取最新的稳定 AWS for Fluent Bit 映像。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:stable
```

> **Note**
>
> 可以进行未经身份验证的拉取，但速率限制低于经过身份验证的拉取。拉取前如果要使用 AWS 账户验证身份，请使用以下命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Windows

Amazon ECR Public Gallery 中 Fluent Bit 映像的 AWS 支持带有以下操作系统的 AMD64 架构：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

AWS Fargate 上的 Windows 容器不支持 FireLens。

通过使用所需的映像标记指定存储库 URL，可以从 Amazon ECR 公共图库中提取 AWS for Fluent Bit 映像。可在 Amazon ECR Public Gallery 上的映像标签选项卡上找到映像标签。

下面说明 Docker CLI 使用的语法。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:tag
```

例如，您可以使用此 Docker CLI 命令拉取最新的稳定 AWS for Fluent Bit 映像。

```
docker pull public.ecr.aws/aws-observability/aws-for-fluent-bit:windowsservercore-
stable
```

> **Note**
>
> 可以进行未经身份验证的拉取，但速率限制低于经过身份验证的拉取。拉取前如果要使用 AWS 账户验证身份，请使用以下命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Amazon ECR

AWS for Fluent Bit 映像可在 Amazon ECR 上获得高可用性。这些映像在大多数AWS 区域可用，包括 AWS GovCloud (US)。

Linux

可以使用以下命令检索最新的稳定 AWS for Fluent Bit 映像 URI。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/stable \
      --region us-east-1
```

使用以下命令来查询 Systems Manager Parameter Store 参数可以列出 AWS for Fluent Bit 映像的所有版本。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit \
      --region us-east-1
```

可以通过引用 Systems Manager 参数存储名称在 AWS CloudFormation 模板中引用最新的稳定 AWS for Fluent Bit 映像。以下是 示例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
```

```
    Default: /aws/service/aws-for-fluent-bit/stable
```

Windows

可以使用以下命令检索最新的稳定 AWS for Fluent Bit 映像 URI。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/windowsservercore-stable \
      --region us-east-1
```

使用以下命令来查询 Systems Manager Parameter Store 参数可以列出 AWS for Fluent Bit 映像的所有版本。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit/windowsservercore \
      --region us-east-1
```

可以通过引用 Systems Manager 参数存储名称在 AWS CloudFormation 模板中引用最新的稳定 AWS for Fluent Bit 映像。以下是 示例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/windowsservercore-stable
```

Amazon ECS 任务定义示例：将日志路由到 FireLens

要将自定义日志路由与 FireLens 结合使用，您必须在任务定义中指定以下内容：

- 包含 FireLens 配置的日志路由器容器。我们建议将容器标记为 `essential`。
- 一个或多个包含指定 `awsfirelens` 日志驱动程序的日志配置的应用程序容器。
- 一个任务 IAM 角色 Amazon 资源名称（ARN），其中包含任务路由日志所需的权限。

使用 AWS Management Console创建新的任务定义时，通过 Firelens 集成部分可以轻松添加日志路由器容器。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

Amazon ECS 将转换日志配置并生成 Fluentd 或 Fluent Bit 输出配置。输出配置挂载在 `/fluent-bit/etc/fluent-bit.conf`（对于 Fluent Bit）和 `/fluentd/etc/fluent.conf`（对于 Fluentd）处的日志路由容器中。

> **Important**
>
> FireLens 在端口 24224 上侦听。因此，为了确保从任务外部无法访问 FireLens 日志路由器，不得允许任务使用的安全组中端口 24224 上的入口流量。对于使用 awsvpc 网络模式的任务，这是与任务关联的安全组。对于使用 host 网络模式的任务，它是与托管任务的 Amazon EC2 实例关联的安全组。对于使用 bridge 网络模式的任务，请勿创建任何使用端口 24224 的端口映射。

默认情况下，Amazon ECS 会在日志条目中添加其他字段来帮助标识日志源。

- `ecs_cluster` - 任务所属的集群的名称。
- `ecs_task_arn` – 容器所属的任务的完整 Amazon 资源名称（ARN）。
- `ecs_task_definition` - 任务正在使用的任务定义名称和修订。
- `ec2_instance_id` – 容器托管于的 Amazon EC2 实例 ID。此字段仅对使用 EC2 启动类型的任务有效。

如果您不想要元数据，可以将 `enable-ecs-log-metadata` 设置为 `false`。

下面的任务定义示例定义了一个日志路由器容器，它使用 Fluent Bit 将日志路由到 CloudWatch Logs。该示例还定义了一个应用程序容器，它使用日志配置将日志路由到 Amazon Data Firehose 并将用于缓冲事件的内存设置为 2MiB。

> **Note**
>
> 有关更多示例任务定义，请参阅 GitHub 上的 Amazon ECS FireLens 示例。

```
{
  "family": "firelens-example-firehose",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
```

```
        "type": "fluentbit",
        "options": {
          "enable-ecs-log-metadata": "true"
        }
      },
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "firelens-container",
          "awslogs-region": "us-west-2",
          "awslogs-create-group": "true",
          "awslogs-stream-prefix": "firelens"
        }
      },
      "memoryReservation": 50
    },
    {
      "essential": true,
      "image": "httpd",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "firehose",
          "region": "us-west-2",
          "delivery_stream": "my-stream",
          "log-driver-buffer-limit": "2097152"
        }
      },
      "memoryReservation": 100
    }
  ]
}
```

在 `logConfiguration` 对象中指定为选项的键值对用于生成 Fluentd 或 Fluent Bit 输出配置。以下是来自 Fluent Bit 输出定义的代码示例。

```
[OUTPUT]
    Name   firehose
    Match  app-firelens*
    region us-west-2
    delivery_stream my-stream
```

> **Note**
>
> FireLens 可管理 `match` 配置。您没有在任务定义中指定 `match` 配置。

使用自定义配置文件

您可以指定自定义配置文件。配置文件格式是您所使用的日志路由器的本机格式。有关更多信息，请参阅 Fluentd Config 文件语法和 Fluent Bit 配置文件。

在您的自定义配置文件中，对于使用 `bridge` 或 `awsvpc` 网络模式的任务，请勿通过 TCP 设置 Fluentd 或 Fluent Bit 转发输入，因为 FireLens 会将它添加到输入配置中。

您的 FireLens 配置必须包含以下选项才能指定自定义配置文件：

`config-file-type`

自定义配置文件的源位置。可用选项为 `s3` 或 `file`。

> **Note**
>
> 托管在 AWS Fargate 上的任务仅支持 `file` 配置文件类型。

`config-file-value`

自定义配置文件的源。如果使用 `s3` 配置文件类型，则配置文件值是 Amazon S3 存储桶和文件的完整 ARN。如果使用 `file` 配置文件类型，则配置文件值是容器映像中或挂载到容器中的卷上存在的配置文件的完整路径。

> **Important**
>
> 在使用自定义配置文件时，必须指定一个与 FireLens 所用路径不同的路径。Amazon ECS 保留对于 Fluent Bit 和 `/fluentd/etc/fluent.conf` 对于 Fluentd 的文件路径 `/fluent-bit/etc/fluent-bit.conf`。

以下示例显示了指定自定义配置时所需的语法。

> **Important**
>
> 要指定托管于 Amazon S3 中的自定义配置文件，请确保已创建具有适当权限的任务执行 IAM 角色。

下面显示了在指定自定义配置时所需的语法。

```
{
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "config-file-type": "s3 | file",
          "config-file-value": "arn:aws:s3:::mybucket/fluent.conf | filepath"
        }
      }
    }
  ]
}
```

> **Note**
>
> 托管在 AWS Fargate 上的任务仅支持 `file` 配置文件类型。

在 Amazon ECS 中使用非 AWS 容器映像

使用私有注册表在 AWS Secrets Manager 中存储您的凭证，然后在任务定义中引用它们。这提供了一种方法来引用存在于 AWS 以外的私有注册表中的容器镜像，这需要在任务定义中进行身份验证。在 Fargate 上托管的任务、Amazon EC2 实例以及使用 Amazon ECS Anywhere 的外部实例都支持此功能。

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

对于 Amazon EC2 实例上托管的任务，此功能要求您具有版本 `1.19.0` 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。有关如何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。

对于 Fargate 上托管的任务，此功能需要平台版本 `1.2.0` 或更高版本。有关信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本。

在容器定义中，使用您创建的密钥的详细信息指定 `repositoryCredentials` 对象。引用的密钥可以来自与使用此密钥的任务不同的 AWS 区域 或不同的账户。

> **Note**
>
> 使用 Amazon ECS API、AWS CLI 或 AWS SDK 时，如果密钥存在于要启动的任务所在的 AWS 区域，可以使用密钥的完整 ARN 或名称。如果密钥存在于另一个账户中，则必须指定密钥的完整 ARN。使用 AWS Management Console 时，必须始终指定密钥的完整 ARN。

下面是显示必需参数的任务定义代码段：

将 *`private-repo`* 替代为私有存储库主机名称，将 *`private-image`* 替代为映像名称。

```
"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。此方法仅支持 Amazon EC2 实例上托管的任务。有关更多信息，请参阅 为私有 Docker 映像配置 Amazon ECS 容器实例 。

要使用私有注册表

1. 任务定义必须具有任务执行角色。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。

   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添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM policy。

   - `secretsmanager:GetSecretValue`
   - `kms:Decrypt` - 仅当密钥使用自定义 KMS 密钥而不是原定设置密钥时才需要。您的自定义密钥的 Amazon 资源名称（ARN）必须添加为资源。

   下面是添加所需权限的示例内联策略。

```
{
    "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"
            ]
        }
    ]
}
```

2. 使用 AWS Secrets Manager 为您的私有注册表凭证创建密钥。有关如何创建密钥的信息，请参阅《AWS Secrets Manager 用户指南》中的创建 AWS Secrets Manager 密钥。

   使用以下格式输入您的私有注册表凭证：

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

3. 注册任务定义。有关更多信息，请参阅 the section called “使用控制台创建任务定义”。

将单个环境变量传递给 Amazon ECS 容器

> **Important**
>
> 我们建议将您的敏感数据存储在 AWS Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数中。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。任务定义中指定的环境变量可由允许执行任务定义 `DescribeTaskDefinition` 操作的所有用户和角色读取。

您可以通过以下方式将环境变量传递到容器：

- 单独，使用 `environment` 容器定义参数。这会将 `--env` 选项映射到 docker run。
- 批量，使用 `environmentFiles` 容器定义参数列出包含环境变量的一个或多个文件。该文件必须托管在 Amazon S3 中。这会将 `--env-file` 选项映射到 docker run。

以下是演示如何指定单个环境变量的任务定义的代码段。

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environment": [
                {
```

```
                    "name": "variable",
                    "value": "value"
                }
            ],
            ...
        }
    ],
    ...
}
```

将环境变量传递给 Amazon ECS 容器

> **Important**
>
> 我们建议将您的敏感数据存储在 AWS Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数中。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。
> 环境变量文件是 Amazon S3 中的对象，所有 Amazon S3 安全注意事项都适用。
> 您不能在 Windows 容器和 Fargate 上的 Windows 容器上使用 `environmentFiles` 参数。

您可以创建环境变量文件，并将其存储在 Amazon S3 中，以便将环境变量传递给您的容器。

通过在文件中指定环境变量，您可以批量注入环境变量。在容器定义中，使用包含环境变量文件的 Amazon S3 存储桶列表指定 `environmentFiles` 对象。

Amazon ECS 不对环境变量强制实施大小限制，但大型环境变量文件可能会填满磁盘空间。使用环境变量文件的每个任务都会导致文件的副本下载到磁盘。作为任务清理的一部分，Amazon ECS 会删除文件。

有关支持的环境变量的信息，请参阅高级容器定义参数 - 环境。

在容器定义中指定环境变量文件时，请考虑以下因素。

- 对于 Amazon EC2 上的 Amazon ECS 任务，您的容器实例需要版本 `1.39.0` 或更高版本的容器代理才能使用此功能。有关如何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。
- 对于 AWS Fargate 上的 Amazon ECS 任务，您的任务必须使用平台版本 `1.4.0` 或更高版本（Linux）来使用此功能。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。

  验证操作系统平台是否支持该变量。有关更多信息，请参阅 the section called “容器定义” 和 the section called “其他任务定义参数”。

- 该文件必须使用 .env 文件扩展名和 UTF-8 编码。
- 每个任务定义最多只能有 10 个文件。
- 环境文件中的每一行都必须包含 VARIABLE=VALUE 格式的环境变量。空格或引号作为 Amazon ECS 文件的值的一部分包含。以开头的行 # 被视为注释并会被忽略。有关环境变量文件语法的更多信息，请参阅在文件中声明默认环境变量。

  以下是合适的语法。

```
#This is a comment and will be ignored
VARIABLE=VALUE
ENVIRONMENT=PRODUCTION
```

- 如果在容器定义中存在使用 environment 参数指定的环境变量，则这些变量的优先级高于环境文件中包含的变量。
- 如果指定了多个环境文件并且其中包含相同变量，则会按输入顺序处理这些文件。这意味着将使用变量的第一个值，并忽略重复变量的后续值。建议您使用唯一的变量名。
- 如果将环境文件指定为容器覆盖，则将使用它。此外，容器定义中指定的任何其他环境文件都将被忽略。
- 以下规则适用于 Fargate 启动类型：
  - 该文件的处理方式与原生 Docker env-file 类似。
  - 不支持 shell 转义处理。
  - 容器入口点解释 VARIABLE 值。

所需的 IAM 权限

使用此功能需要 Amazon ECS 任务执行角色。这允许容器代理从 Amazon S3 中提取环境变量文件。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

要提供对您创建的 Amazon S3 对象的访问权限，请将以下权限作为内联策略手动添加到任务执行角色。可以使用 Resource 参数将权限范围扩展到包含环境变量文件的 Amazon S3 存储桶。有关更多信息，请参阅添加和删除 IAM policy。

- s3:GetObject
- s3:GetBucketLocation

在以下示例中，内联策略会添加这些权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/env_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

示例

以下是演示如何指定环境变量文件的任务定义的代码段。

```
{
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environmentFiles": [
                {
                    "value": "arn:aws:s3:::s3_bucket_name/envfile_object_name.env",
                    "type": "s3"
                }
            ],
            ...
        }
```

```
    ],
    ...
}
```

将敏感数据传递给 Amazon ECS 容器

您可以将敏感数据（例如数据库凭证）安全地传递到容器中。

您可以使用 Secrets Manager 或以 Systems Manager Parameter Store 中的参数来存储密钥。

您可以通过编程方式从应用程序中检索密钥，也可以使用环境变量。

首先，将敏感数据作为密钥存储在 Secrets Manager 中，或作为参数存储在 Systems Manager Parameter Store 中。然后，使用以下任一方式将密钥公开给容器。

主题

- 在 Amazon ECS 中是密钥管理最佳实践
- 在 Amazon ECS 中以编程方式检索 Secrets Manager 密钥
- 在 Amazon ECS 中以编程方式检索 Systems Manager Parameter Store 密钥
- 通过 Amazon ECS 环境变量检索 Secrets Manager 密钥
- 通过 Amazon ECS 环境变量检索 Systems Manager 参数
- 检索 Amazon ECS 日志记录配置的密钥
- 在 Amazon ECS 中指定使用 Secrets Manager 密钥的敏感数据

在 Amazon ECS 中是密钥管理最佳实践

应用程序经常使用诸如 API 密钥和数据库凭证之类的密钥来访问其他系统。它们通常由用户名和密码、证书或 API 密钥组成。对这些密钥的访问应仅限于使用 IAM 并在运行时注入到容器中的特定 IAM 主体。

密钥可以从 AWS Secrets Manager 和 Amazon EC2 Systems Manager Parameter Store 无缝注入容器中。这些密钥可以在您的任务中作为以下任何内容引用。

1. 它们被引用为使用 `secrets` 容器定义参数的环境变量。
2. 如果您的日志记录平台需要身份验证，它们被引用为 `secretOptions`。有关更多信息，请参阅日志记录配置选项。

3. 如果从中提取容器的注册表需要身份验证，则使用 `repositoryCredentials` 容器定义参数的映像将它们作为密钥引用。从 Amazon ECR Public Gallery 中提取映像时使用此方法。有关更多信息，请参阅任务的私有注册表身份验证。

密钥建议

在设置密钥管理时，建议您执行以下操作。

使用 AWS Secrets Manager Amazon EC2 Systems Manager Parameter Store 来存储密钥材料

您应该将 API 密钥、数据库凭证和其他密钥材料安全地存储在 AWS Secrets Manager 中，或者将其作为加密参数存储在 Amazon EC2 Systems Manager Parameter Store 中。这些服务之所以相似，是因为它们都是使用 AWS KMS 加密敏感数据的托管键值存储。但是，AWS Secrets Manager 还包括自动轮换密钥、生成随机密钥和跨 AWS 账户共享密钥的功能。如果您认为这些功能很重要，请使用 AWS Secrets Manager 以其他方式使用加密参数。

> **Note**
>
> 引用来自 AWS Secrets Manager 或 Amazon EC2 Systems Manager Parameter Store 的密钥的任务需要一个任务执行角色，其具有授予 Amazon ECS 访问所需密钥以及用于加密和解密该密钥的 AWS KMS 密钥（如果适用）的策略。

> **Important**
>
> 任务中引用的密钥不会自动轮换。如果您的密钥发生更改，则必须强制进行新的部署或启动新任务以检索最新的密钥值。有关更多信息，请参阅以下主题：
>
> - AWS Secrets Manager：将数据作为环境变量注入
> - Amazon EC2 Systems Manager Parameter Store：将数据作为环境变量注入

从加密的 Amazon S3 存储桶中检索数据

由于环境变量的值可能会无意中泄漏到日志中并在运行 `docker inspect` 时被显示，因此您应该将密钥存储在加密的 Amazon S3 存储桶中，并使用任务角色来限制对这些密钥的访问。执行此操作时，必须编写应用程序以从 Amazon S3 存储桶中读取密钥。有关说明，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

使用 sidecar 容器将密钥挂载到卷上

由于环境变量会增加数据泄露的风险，因此您应该运行一个 sidecar 容器，以从 AWS Secrets Manager 中读取密钥并将其写入共享卷中。通过使用 Amazon ECS 容器排序，此容器可以在应用程序容器之前运行和退出。当您执行此操作时，应用程序容器随后会挂载写入密钥的卷。与 Amazon S3 存储桶方法类似，必须编写您的应用程序以从共享卷中读取密钥。由于该卷的作用域仅限于任务，因此该卷将在任务停止后自动删除。有关 sidecar 容器的示例，请参阅 aws-secret-sidecar-injector 项目。

> **Note**
>
> 在 Amazon EC2 上，可使用 AWS KMS 客户托管密钥对写入密钥的卷进行加密。在 AWS Fargate 上，使用服务托管密钥自动加密卷存储。

其他 资源

- 在 Amazon ECS 任务中将密钥传递给容器
- Chamber 是在 Amazon EC2 Systems Manager Parameter Store 中存储密钥的包装器

在 Amazon ECS 中以编程方式检索 Secrets Manager 密钥

使用 Secrets Manager 保护敏感数据，并在数据库凭证、API 键和其他密钥的整个生命周期内对其进行轮换、管理和检索。

您可以使用 Secrets Manager 来存储敏感数据，而不是在应用程序中以纯文本形式对敏感信息进行硬编码。

我们建议使用这种方法来检索敏感数据，因为如果随后更新 Secrets Manager 密钥，应用程序会自动检索该密钥的最新版本。

可以在 Secrets Manager 中创建秘密。创建 Secrets Manager 密钥后，更新应用程序代码以检索该密钥。

在 Secrets Manager 中保护敏感数据之前，请查看以下注意事项。

- 仅存储文本数据的机密，这些机密是使用 `SecretString` 参数 CreateSecret 支持 API。不支持存储二进制数据的密钥，这些密钥是使用 CreateSecret API 的 `SecretBinary` 参数创建的。
- 使用接口 VPC 端点增强安全控制。您必须为 Secrets Manager 创建接口 VPC 端点。有关 VPC 端点的信息，请参阅《AWS Secrets Manager 用户指南》中的创建 VPC 端点。

- 您的任务使用的 VPC 必须使用 DNS 解析。

所需的 IAM 权限

要使用此功能，您必须具有 Amazon ECS 任务角色，并在任务定义中引用它。有关更多信息，请参阅 Amazon ECS 任务 IAM 角色。

要提供对您创建的 Secrets Manager 密钥的访问权限，请将以下权限手动添加到任务执行角色。有关如何管理权限的信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

- `secretsmanager:GetSecretValue` – 在引用 Secrets Manager 密钥时是必需的。添加从 Secrets Manager 中检索密钥的权限。

以下示例策略添加了所需的权限。

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

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。有关如何创建密钥的信息，请参阅《AWS Secrets Manager 用户指南》中的创建 AWS Secrets Manager 密钥。

更新应用程序以通过编程方式检索 Secrets Manager 密钥

您可以直接从应用程序调用 Secrets Manager API 来检索密钥。有关信息，请参阅《AWS Secrets Manager 用户指南》中的从 AWS Secrets Manager 中检索密钥。

要检索存储在 AWS Secrets Manager 中的敏感数据，请参阅 AWS SDK 代码示例代码库中的使用 AWS SDK 的 AWS Secrets Manager 的代码示例。

在 Amazon ECS 中以编程方式检索 Systems Manager Parameter Store 密钥

Systems Manager Parameter Store 提供了密钥的安全存储和管理。您可以将密码、数据库字符串、EC2 实例 ID 和 AMI ID 以及许可证代码等数据存储为参数值。可以将值存储为纯文本或加密数据。

您可以使用 Secrets Manager 来存储敏感数据，而不是在应用程序中以纯文本形式对敏感信息进行硬编码。

建议使用这种方法来检索敏感数据，因为如果随后更新 Systems Manager Parameter Store 参数，应用程序会自动检索最新版本。

可以在 Secrets Manager 中创建秘密。创建 Secrets Manager 密钥后，更新应用程序代码以检索该密钥。

在 Systems Manager Parameter Store 中保护敏感数据之前，请查看以下注意事项。

- 仅支持存储文本数据的密钥。不支持存储二进制数据的密钥。
- 使用接口 VPC 端点增强安全控制。
- 您的任务使用的 VPC 必须使用 DNS 解析。

所需的 IAM 权限

要使用此功能，您必须具有 Amazon ECS 任务角色，并在任务定义中引用它。这允许容器代理提取必要的 Systems Manager 资源。有关更多信息，请参阅 Amazon ECS 任务 IAM 角色。

> **Important**
>
> 对于使用 EC2 启动类型的任务，必须使用 ECS 代理配置变量 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` 才能使用此功能。您可以在创建容器实例的过程中将其添加到 `./etc/ecs/ecs.config` 文件中，也可以将其添加到现有实例中，然后重新启动 ECS 代理。有关更多信息，请参阅 Amazon ECS 容器代理配置。

要提供对您创建的 Systems Manager Parameter Store 参数的访问权限，请将以下权限作为策略手动添加到任务执行角色。有关如何管理权限的信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

- `ssm:GetParameters` — 当您在任务定义中引用 Systems Manager Parameter Store 参数时是必需的。添加检索 Systems Manager 参数的权限。
- `secretsmanager:GetSecretValue` — 当您直接引用 Secrets Manager 密钥或者您的 System Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必需的。添加从 Secrets Manager 中检索密钥的权限。
- `kms:Decrypt` — 仅当您的密钥使用客户托管键而不是默认键时才需要。您的自定义密钥的 ARN 应添加为资源。添加解密客户托管密钥的权限。

以下示例策略添加了所需的权限：

```
{
  "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 参数

您可以使用 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》中的创建 Systems Manager 参数（控制台）或创建 Systems Manager 参数 (AWS CLI)。

更新应用程序以通过编程方式检索 Systems Manager Parameter Store 密钥

要检索存储在 Systems Manager Parameter Store 参数中存储的敏感数据，请参阅 AWS SDK 代码示例代码库中的使用 AWS SDK 的 Systems Manager 的代码示例。

通过 Amazon ECS 环境变量检索 Secrets Manager 密钥

将密钥注入为环境变量时，可以指定密钥的完整内容、密钥中的特定 JSON 密钥或要注入的密钥的特定版本。这将帮助您控制提供给容器的敏感数据。有关密钥版本控制的更多信息，请参阅《AWS Secrets Manager 用户指南》中的 AWS Secrets Manager 的主要术语和概念。

在使用环境变量将 Secrets Manager 密钥注入容器时，应考虑以下事项。

- 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，则容器将不会自动接收更新后的值。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。
- 对于 AWS Fargate 上的 Amazon ECS 任务，应注意以下事项：
  - 要将密钥的完整内容注入为环境变量或注入到日志配置中，您必须使用版本 `1.3.0` 或更高版本的平台。有关信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本。
  - 要将特定 JSON 密钥或密钥版本注入为环境变量或注入到日志配置中，您必须使用平台版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0`（Windows）。有关信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本。
- 对于 EC2 上的 Amazon ECS 任务，应注意以下事项：
  - 要使用特定的 JSON 密钥或密钥版本注入密钥，容器实例必须具有版本 `1.37.0` 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。

    要将密钥的完整内容注入为环境变量或将密钥注入日志配置中，您的容器实例必须具有版本 `1.22.0` 或更高版本的容器代理。
- 使用接口 VPC 端点增强安全控制措施，并通过私有子网连接到 Secrets Manager。您必须为 Secrets Manager 创建接口 VPC 端点。有关 VPC 端点的信息，请参阅《AWS Secrets Manager 用户指南》中的创建 VPC 端点。有关使用 Secrets Manager 和 Amazon VPC 的更多信息，请参阅如何在 Amazon VPC 内连接到 Secrets Manager 服务。
- 对于配置为使用 `awslogs` 日志记录驱动程序的 Windows 任务，您还必须在容器实例上设置 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 环境变量。可使用以下语法对用户数据执行此操作：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
```

```
</powershell>
```

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。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

要提供对您创建的 Secrets Manager 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 policy。

- `secretsmanager:GetSecretValue`–在引用 Secrets Manager 密钥时需要。添加从 Secrets Manager 中检索密钥的权限。
- `kms:Decrypt` – 仅当您的密钥使用客户托管键而不是默认键时才需要。您的客户托管键的 ARN 应添加为资源。添加解密客户托管密钥的权限。

以下示例策略添加了所需的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "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 AWS Secrets Manager 密钥。

将环境变量添加到容器定义中

在容器定义中，可以指定以下内容：

- `secrets` 对象，该对象包含要在容器中设置的环境变量名称
- Secrets Manager 密钥的 Amazon Resource Name (ARN)。
- 其他参数，这些参数包含要提供给容器的敏感数据

以下示例显示必须为 Secrets Manager 密钥指定的完整语法。

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

下一部分介绍了其他参数。这些参数是可选的，但如果不使用它们，则必须包含冒号 : 以使用默认值。下面提供了示例，以便您了解更多上下文。

`json-key`

使用要设置为环境变量值的值指定密钥-值对中密钥的名称。仅支持 JSON 格式的值。如果未指定 JSON 密钥，则使用密钥的完整内容。

`version-stage`

指定要使用的密钥版本的暂存标签。如果指定了版本的暂存标签，则无法指定版本 ID。如果未指定版本阶段，则默认行为是使用 `AWSCURRENT` 暂存标签检索密钥。

暂存标签用于在更新或轮换密钥的各个版本时对其进行跟踪。密钥的每个版本均有一个或多个暂存标签和一个 ID。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的 AWS Secrets Manager 主要术语和概念。

`version-id`

指定要使用的密钥版本的唯一标识符。如果指定了版本 ID，则无法指定版本暂存标签。如果未指定版本 ID，则默认行为是使用 `AWSCURRENT` 暂存标签检索密钥。

版本 ID 用于在更新或轮换密钥的各个版本时对其进行跟踪。密码的每个版本均有一个 ID。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的 AWS Secrets Manager 主要术语和概念。

示例容器定义

以下示例说明了可用于在容器定义中引用 Secrets Manager 密钥的方法。

Example 引用完整密钥

以下是任务定义的片段，其中显示引用 Secrets Manager 密钥的完全文本时的格式。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

要从容器中访问此密钥的值，需要调用 $environment_variable_name。

Example 引用密钥中的特定密钥

下面显示了 get-secret-value 命令中的示例输出，其中显示了密钥的内容以及与之关联的版本暂存标签和版本 ID。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981ddEXAMPLE",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定密钥。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::"
    }]
```

```
  }]
}
```

Example 引用特定的密钥版本

下面显示了 describe-secret 命令中的示例输出，其中显示了密钥的未加密内容以及密钥的所有版本的元数据。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981ddEXAMPLE": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE": [
            "AWSPREVIOUS"
        ]
    }
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定版本暂存标签。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定版本 ID。

```
{
  "containerDefinitions": [{
```

```
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Example 引用密钥的特定密钥和版本暂存标签

以下内容说明如何同时引用密钥中的特定密钥和特定版本暂存标签。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

要指定特定密钥和版本 ID，请使用以下语法。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

有关如何使用环境变量中指定的密钥创建任务定义的信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

通过 Amazon ECS 环境变量检索 Systems Manager 参数

您可以使用 Amazon ECS 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。

在使用环境变量将 Systems Manager 密钥注入容器时，请考虑以下事项。

- 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，则容器将不会自动接收更新后的值。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。
- 对于 AWS Fargate 上的 Amazon ECS 任务，应注意以下事项：
  - 要将密钥的完整内容注入为环境变量或注入到日志配置中，您必须使用版本 `1.3.0` 或更高版本的平台。有关信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本。
  - 要将特定 JSON 密钥或密钥版本注入为环境变量或注入到日志配置中，您必须使用平台版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0`（Windows）。有关信息，请参阅适用于 Amazon ECS 的 Fargate Linux 平台版本。
- 对于 EC2 上的 Amazon ECS 任务，应注意以下事项：
  - 要使用特定的 JSON 密钥或密钥版本注入密钥，容器实例必须具有版本 `1.37.0` 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。

    要将密钥的完整内容注入为环境变量或将密钥注入日志配置中，您的容器实例必须具有版本 `1.22.0` 或更高版本的容器代理。
- 使用接口 VPC 端点增强安全控制。您必须为 Systems Manager 创建接口 VPC 端点。有关 VPC 端点的信息，请参阅《AWS Systems Manager 用户指南》中的创建 VPC 端点。
- 对于配置为使用 `awslogs` 日志记录驱动程序的 Windows 任务，您还必须在容器实例上设置 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 环境变量。可使用以下语法对用户数据执行此操作：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

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 Systems Manager 资源。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

> **Important**
>
> 对于使用 EC2 启动类型的任务，必须使用 ECS 代理配置变量 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` 才能使用此功能。您可以在创建容器实例的过程中将其添加到 `./etc/ecs/ecs.config` 文件中，也可以将其添加到现有实例中，然后重新启动 ECS 代理。有关更多信息，请参阅 Amazon ECS 容器代理配置。

要提供对您创建的 Systems Manager Parameter Store 参数的访问权限，请将以下权限手动添加到任务执行角色。有关如何管理权限的信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

- `ssm:GetParameters` — 当您在任务定义中引用 Systems Manager Parameter Store 参数时是必需的。添加检索 Systems Manager 参数的权限。
- `secretsmanager:GetSecretValue` — 当您直接引用 Secrets Manager 密钥或者您的 System Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必需的。添加从 Secrets Manager 中检索密钥的权限。
- `kms:Decrypt` — 仅当您的密钥使用客户托管键而不是默认键时才需要。您的自定义密钥的 ARN 应添加为资源。添加解密客户托管密钥的权限。

以下示例策略添加了所需的权限：

```
{
  "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 Systems Manager 参数

您可以使用 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》中的创建 Systems Manager 参数（控制台）或创建 Systems Manager 参数 (AWS CLI)。

将环境变量添加到容器定义中

在容器定义中，使用要在容器中设置的环境变量的名称和包含要提供给容器的敏感数据的 Systems Manager Parameter Store 参数的完整 ARN 指定 `secrets`。有关更多信息，请参阅 `secrets`。

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。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，则指定完整的 ARN。

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

有关如何使用环境变量中指定的密钥创建任务定义的信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

检索 Amazon ECS 日志记录配置的密钥

您可以使用 `logConfiguration` 中的 `secretOptions` 参数来传递用于日志记录的敏感数据。

您可以将密钥存储在 Secrets Manager 或 Systems Manager 中。

使用 Secrets Manager

在容器定义中，当指定 `logConfiguration` 时，您可以同时指定 `secretOptions`，方法是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 密钥的完整 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://your_splunk_instance:8088"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

使用 Systems Manager

您可以在日志配置中注入敏感数据。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 Systems Manager Parameter Store 参数的完整 ARN 指定 `secretOptions`。

> **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，则指定完整的 ARN。

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"
      }]
    }]
  }]
}
```

在 Amazon ECS 中指定使用 Secrets Manager 密钥的敏感数据

使用 Amazon ECS 时，您可以将敏感数据存储在 AWS Secrets Manager 密钥中，然后在容器定义中引用这些密钥，从而将敏感数据注入容器。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。

了解如何创建 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：

- 设置以使用 Amazon ECS 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。

key/value pairs to be stored in this secret（要存储在此密钥中的键/值对）是教程末尾处容器中的环境变量值。

保存密钥 ARN 以便在后续步骤中在任务执行 IAM policy 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。

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，选择角色。

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"
            ]
        }
    ]
}
```

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

可以使用 Amazon ECS 控制台创建引用了 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/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。
3. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
4. 在 JSON 编辑器框中，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。选择保存。
5.

```
{
```

```
    "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",
                    "name": "username_value"
                }
            ],
            "memory": 300,
            "image": "httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

6. 选择创建。

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，我们将创建一个 `t2.micro` 容器实例，使用经 Amazon ECS 优化的 Amazon Linux 2 AMI。

有关如何为 EC2 启动类型创建集群的信息，请参阅 the section called “为 Amazon EC2 启动类型创建集群”。

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。

有关如何运行任务的信息，请参阅 the section called “将应用程序作为任务运行”。

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/v2 打开控制台。
   b. 在导航窗格中，选择集群，然后选择您创建的集群。
   c. 选择基础设施，然后选择容器实例。
   d. 记录您的实例的公有 IP 或公有 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》中的使用 PuTTY 从 Windows 连接到 Linux 实例。

> **Important**
>
> 要详细了解您在连接到实例时遇到的任何问题，请参阅《Amazon EC2 用户指南》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：清理

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/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 选择 Delete Cluster (删除集群)。
5. 在确认框中，输入 delete ***cluster name***，然后选择删除。

6. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
7. 在导航窗格中，选择角色。
8. 在角色列表中搜索 ecsTaskExecutionRole 并将其选定。
9. 选择权限，然后选择 ECSSecretsTutorial 旁边的 X。选择移除。
10. 打开 Secrets Manager 控制台，网址为 https://console.aws.amazon.com/secretsmanager/。
11. 选择您之前创建的 username_value 密钥，然后选择操作、删除密钥。

Amazon ECS 任务定义参数

任务定义包括以下各部分：任务系列、AWS Identity and Access Management（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。任务定义中需要系列和容器定义。相反，任务角色、网络模式、卷、任务放置约束和启动类型是可选的。

您可以在 JSON 文件中使用这些参数来配置任务定义。

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。

启动类型

您在注册任务定义时，可以指定 Amazon ECS 应验证任务定义的启动类型。如果任务定义未根据指定的兼容性进行验证，则返回客户端异常。有关更多信息，请参阅 Amazon ECS 启动类型。

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 | EXTERNAL

验证任务定义的任务启动类型。这将开始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。

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 Amazon ECS 任务 IAM 角色。

在您启动经 Amazon ECS 优化的 Windows Server AMI 时，Windows 上任务的 IAM 角色需要设置 `-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 Amazon EC2 Windows 实例附加配置。

任务执行角色

`executionRoleArn`

类型：字符串

必需：条件

任务执行角色的 Amazon 资源名称（ARN），该角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。

> **Note**
>
> 任务执行 IAM 角色是必需的，具体取决于任务的要求。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

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。对于 Amazon EC2 Linux 实例上托管的 Amazon ECS 任务，有效值为 `none`、`bridge`、`awsvpc` 和 `host`。如果未指定网络模式，原定设置网络模式为`bridge`。对于在 Amazon EC2 Windows 实例上托管的 Amazon ECS 任务，有效值为 `default` 和`awsvpc`。如果未指定网络模式，则 `default` 使用网络模式。对于 Fargate 托管的 Amazon ECS 任务，要求使用 `awsvpc` 网络模式。

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`，则任务在 Linux 上使用 Docker 的内置虚拟网络，该网络在托管任务的每个 Amazon EC2 实例内运行。Linux 上的内置虚拟网络使用 `bridge` Docker 网络驱动程序。

如果网络模式为 `host`，任务将通过直接映射容器端口到托管任务的 Amazon EC2 实例的 ENI，使用会绕过 Docker 内置虚拟网络的主机网络。动态端口映射不可在此网络模式中使用。使用此模式的任务定义中的容器必须指定具体的 `hostPort` 号。主机上的端口号不可被用于多个任务。因此，您不可以在单个 Amazon EC2 实例上运行相同任务定义的多个任务。

> **Important**
>
> 运行使用 `host` 网络模式的任务时，请勿使用根用户（UID 0）运行容器，以获得更好的安全性。作为安全最佳实践，请始终使用非根用户。

对于 Amazon EC2 启动类型，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 EC2 启动类型的 Amazon ECS 任务联网选项。

如果网络模式为 `default`，则任务在 Windows 上使用 Docker 的内置虚拟网络，该网络在托管任务的每个 Amazon EC2 实例内运行。Windows 上的内置虚拟网络使用 `nat` Docker 网络驱动程序。

对于 Fargate 启动类型，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

Fargate 任务联网。awsvpc 网络模式可为容器提供最高联网性能，因为容器使用 Amazon EC2 网络堆栈。公开的容器端口直接映射到附加的弹性网络接口端口。因此，您不能使用动态主机端口映射。

host 和 awsvpc 网络模式可为容器提供最高联网性能，因为容器使用 Amazon EC2 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，您不能使用动态主机端口映射。

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。如果 Windows，default 和 awsvpc 模式可以使用。

运行时平台

operatingSystemFamily

类型：字符串

必需：条件

原定设置：LINUX

对于在 Fargate 上托管的 Amazon ECS 任务，此参数是必需的。

当您注册任务定义时，您可以指定操作系统系列。

托管在 Fargate 上的 Amazon ECS 任务的有效值为 LINUX、WINDOWS_SERVER_2019_FULL、WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2022_ 和 WINDOWS_SERVER_2022_CORE。

托管在 EC2 上的 Amazon ECS 任务的有效值为 LINUX、WINDOWS_SERVER_2022_CORE、WINDOWS_SERVER_2022_FULL、WINDOWS_SERVER_2019_ 以及 WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2016_FULL、WINDOWS_SERVER_2004_CORE 和 WINDOWS_SERVER_20H2_CORE。

服务中使用的所有任务定义对于此参数都必须具有相同的值。

当任务定义是服务的一部分时，此值必须与服务 platformFamily 值匹配。

`cpuArchitecture`

类型：字符串

必需：条件

原定设置：X86_64

对于在 Fargate 上托管的 Amazon ECS 任务，此参数是必需的。如果将参数保留为 `null`，则在 Fargate 上托管的任务启动时会自动分配默认值。

当您注册任务定义时，您可以指定 CPU 架构。有效值为 `X86_64` 和 `ARM64`。

服务中使用的所有任务定义对于此参数都必须具有相同的值。

当您有 Fargate 启动类型或 EC2 启动类型的 Linux 任务时，可以将值设置为 `ARM64`。有关更多信息，请参阅 the section called “适用于 64 位 ARM 工作负载的任务定义”。

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。对于 Amazon EC2 实例上托管的任务，这些字段是可选的。对于 Fargate 上托管的任务（Linux 和 Windows），这些字段是必填字段，支持的 `cpu` 和 `memory` 有具体值。

> **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：条件

> **Note**
>
> Windows 容器不支持此参数。

要为任务提供的 CPU 单位的硬限制。您可以通过 CPU 单位数或虚拟 CPU（vCPU）数字符串的方式在 JSON 文件中将 CPU 值。例如，您可以使用 `1024`（CPU 单位数）或 `1 vCPU`（vCPU 数）来指定 CPU 值。注册任务定义时，vCPU 值将转换为指示 CPU 单元的整数。

对于 EC2 实例或外部实例上运行的任务，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。在 EC2 或外部实例上运行的任务支持的值介于 `0.125` 个 vCPU 与 `10` 个 vCPU 之间。

对于 Fargate 上托管的任务（Linux 和 Windows 容器），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。下表显示了任务级 CPU 和内存的有效组合。

CPU 值	内存值	AWS Fargate 支持的操作系统
256 (.25 vCPU)	512MiB、1GB、2GB	Linux
512 (.5 vCPU)	1GB、2GB、3GB、4GB	Linux
1024 (1 vCPU)	2GB、3GB、4 GB、5GB、6G B、7GB、8GB	Linux、Windows
2048 (2 vCPU)	4GB 到 16GB 之间 (以 1GB 为增量)	Linux、Windows
4096 (4 vCPU)	8GB 到 30GB 之间 (以 1GB 为增量)	Linux、Windows
8192 (8 vCPU) **Note** 此选项需要 Linux 平台 `1.4.0` 或更高版本。	16 GB 到 60 GB 之间（以 4 GB 为增量）	Linux
16384 (16vCPU)	32 GB 到 120 GB 之间（以 8 GB 为增量）	Linux

CPU 值	内存值	AWS Fargate 支持的操作系统
Note 此选项需要 Linux 平台 1.4.0 或更高版本。		

`memory`

类型：字符串

必需：条件

> **Note**
>
> Windows 容器不支持此参数。

要向任务提供的内存硬限制。您可以通过以二进制兆字节（MiB）或千兆字节（GB）字符串方式在任务定义中指定内存值。例如，您可以使用 3072（以 MiB 为单位）或 3 GB（以 GB 为单位）指定内存值。注册任务定义时，GB 值将转换为指示 MiB 的整数。

对于 Amazon EC2 实例上托管的任务，此字段是可选的，可使用任何值。如果指定了任务级内存值，则容器级内存值是可选的。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。您可以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。有关更多信息，请参阅 预留 Amazon ECS Linux 容器实例内存 。

对于 Fargate 上托管的任务（Linux 和 Windows 容器），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（以 MiB 为单位，近似等效值以 GB 为单位）	CPU 值	Fargate 支持的操作系统
512 (0.5GB)、1024 (1GB)、2048 (2GB)	256 (.25 vCPU)	Linux

内存值（以 MiB 为单位，近似等效值以 GB 为单位）	CPU 值	Fargate 支持的操作系统
1024 (1GB)、2048 (2GB)、3072 (3GB)、4096 (4GB)	512 (.5 vCPU)	Linux
2048 (2GB)、3072 (3GB)、4096 (4GB)、5120 (5GB)、6144 (6GB)、7168 (7GB)、8192 (8GB)	1024 (1 vCPU)	Linux、Windows
4096 (4GB) 到 16384 (16GB) 之间（以 1024 (1GB) 为增量）	2048 (2 vCPU)	Linux、Windows
8192 (8GB) 到 30720 (30GB) 之间（以 1024 (1GB) 为增量）	4096 (4 vCPU)	Linux、Windows
16 GB 到 60 GB 之间（以 4 GB 为增量） **Note** 此选项需要 Linux 平台 `1.4.0` 或更高版本。	8192 (8 vCPU)	Linux
32 GB 到 120 GB 之间（以 8 GB 为增量） **Note** 此选项需要 Linux 平台 `1.4.0` 或更高版本。	16384 (16vCPU)	Linux

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 进程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
- 高级容器定义参数
- 其他容器定义参数

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
- 图像
- 内存
- 端口映射
- 私有存储库凭证

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，则可在另一个容器的 links 中输入一个容器的 name。这样是为了连接容器。

图像

`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 进程守护程序。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。此参数可映射到 Docker Remote API 的创建容器部分中的 `Image` 和 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 在 Amazon ECS 中使用非 AWS 容器映像。
- 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。

如果您使用的是 Fargate 启动类型，则此参数是可选参数。

如果您使用的是 EC2 启动类型，则必须指定任务级内存值或容器级内存值。如果同时指定容器级 memory 和 memoryReservation 值，则 memory 值必须大于 memoryReservation 值。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 20.10.0 或更高版本进程守护程序将为容器预留最少 6MiB 的内存。因此，不要为容器指定少于 6MiB 的内存。

Docker 19.03.13-ce 或更早版本的进程守护程序将为容器预留最少 4MiB 的内存。因此，不要为容器指定少于 4MiB 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，请参阅预留 Amazon ECS Linux 容器实例内存 。

`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范围内。但是，您的容器在需要时可使用更多内存。容器可以用尽使用 memory 参数指定的硬限制（如果适用）或容器实例中的全部可用内存，以较低者为准。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。

如果未指定任务级内存值，则必须为容器定义中的一个或两个 memory 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。

例如，假设您的容器通常使用 128MiB 内存，但有时会在短时间内迸发至 256MiB 内存。您可以将 memoryReservation 设置为 128MiB，将 memory 硬限制设置为 300 MiB。此配置允许容器从容器实例上的剩余资源中预留 128MiB 内存。同时，此配置还允许容器在需要时使用更多内存资源。

> **Note**
>
> Windows 容器不支持此参数。

Docker 20.10.0 或更高版本进程守护程序将为容器预留最少 6MiB 的内存。因此，不要为容器指定少于 6MiB 的内存。

Docker 19.03.13-ce 或更早版本的进程守护程序将为容器预留最少 4MiB 的内存。因此，不要为容器指定少于 4MiB 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，请参阅预留 Amazon ECS Linux 容器实例内存 。

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，只能指定 `containerPort`。`hostPort` 可以留空或值与 `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。

该参数的大部分字段（包括 `containerPort`、`hostPort`、`protocol`）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。
- 元数据：任务元数据端点。

`appProtocol`

类型：字符串

必需：否

用于端口映射的应用程序协议。此参数仅适用于服务连接。我们建议您将此参数设置为与应用程序使用的协议一致。如果您设置此参数，Amazon ECS 会将协议特定的连接处理添加到服务连接代理。如果您设置此参数，Amazon ECS 会在 Amazon ECS 控制台和 CloudWatch 中添加协议特定的遥测。

如果您没有为此参数设置值，则使用 TCP。但是，Amazon ECS 不会为 TCP 添加协议特定的遥测。

有关更多信息，请参阅 the section called “ Service Connect”。

有效的协议值：`"HTTP" | "HTTP2" | "GRPC"`

`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 启动类型的任务中使用容器，公开的端口必须使用 `containerPort` 指定。

对于 Fargate 上的 Windows 容器，不能将端口 3150 用于 `containerPort`。这是因为它是预留的。

假设您正在具有 EC2 启动类型的任务中使用容器，且指定了容器端口，而非主机端口。然后，您的容器将自动接收临时端口范围内的主机端口。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 个预留端口配额。

`containerPortRange`

类型：字符串

必需：否

容器上的端口号范围，该容器绑定到动态映射的主机端口范围。

您只能使用 `register-task-definition` API 设置此参数。该选项在 `portMappings` 参数中可用。有关更多信息，请参阅《AWS Command Line Interface 参考》中的 register-task-definition。

当您指定 `containerPortRange` 时，以下规则适用：

- 必须使用 `bridge` 网络模式或 `awsvpc` 网络模式。
- 此参数同时适用于 EC2 和 AWS Fargate 启动类型。
- 此参数同时适用于 Linux 和 Windows 操作系统。
- 容器实例必须拥有不低于 1.67.0 版本的容器代理和不低于 1.67.0-1 版本的 `ecs-init` 程序包。
- 每个容器最多可指定 100 个端口范围。
- 您不用指定 `hostPortRange`。`hostPortRange` 的值设置如下：
  - 对于使用 `awsvpc` 网络模式的任务中的容器，`hostPort` 将设置为与 `containerPort` 相同的值。这是一种静态映射策略。
  - 对于使用 `bridge` 网络模式的任务中的容器，Amazon ECS 代理会从默认临时范围内找到开放主机端口，并将其传递给 Docker 以将其绑定到容器端口。
- `containerPortRange` 有效值为 1 到 65535。
- 每个容器的一个端口只能包含在的一个端口映射中。
- 您不能指定重叠的端口范围。
- 范围中的第一个端口必须小于范围中的最后一个端口。
- 当有大量端口时，Docker 建议您关闭 Docker 进程守护程序配置文件中的 docker-proxy。

  有关更多信息，请参阅 Github 上的问题 #11185。

  有关如何关闭 Docker 进程守护程序配置文件中的 docker-proxy 的信息，请参阅《Amazon ECS 开发者指南》中的 Docker 进程守护程序。

您可以调用 `DescribeTasks` 以查看 `hostPortRange`，它们是绑定到容器端口的主机端口。

该端口范围不包含在发送到 EventBridge 的 Amazon ECS 任务事件中。有关更多信息，请参阅 the section called “使用 EventBridge 自动响应 Amazon ECS 错误”。

`hostPortRange`

类型：字符串

必需：否

用于网络绑定的主机上的端口号范围。它由 Docker 分配并由 Amazon ECS 代理传送。

`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。

假设您正在具有 EC2 启动类型的任务中使用容器。您可以为容器端口映射指定非预留主机端口。此项称为静态主机端口映射。或者，您可以在指定 `containerPort` 时省略 `hostPort`（或将其设置为 `0`）。您的容器会自动接收容器实例操作系统和 Docker 版本的临时端口范围内的端口。这称为动态主机端口映射。

Docker 版本 1.6.0 及更高版本的默认临时端口范围在 `/proc/sys/net/ipv4/ip_local_port_range` 下的实例上列出。如果此内核参数不可用，则将使用来自 49153–65535 的原定设置临时端口范围。不要尝试指定临时端口范围内的主机端口。这是因为这些都是为自动分配预留的。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 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。任务停止后，系统会释放主机端口。当前预留端口将显示在 describe-container-instances 输出的 `remainingResources` 中。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 个预留端口配额。

`name`

类型：字符串

必填项：否，在服务中配置 Service Connect 时需要

用于端口映射的名称。此参数仅适用于服务连接。此参数是您在服务的 Service Connect 配置中使用的名称。

有关更多信息，请参阅 使用 Service Connect 连接具有短名称的 Amazon ECS 服务。

在以下示例中，使用了针对 Service Connect 的两个必填字段。

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
]
```

`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**
>
> Service Connect 仅支持 `tcp`。请记住，如果未设置此字段，则表示 `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。

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
    }
    ...
]
```

私有存储库凭证

`repositoryCredentials`

类型：RepositoryCredentials 对象

必需：否

私有注册表身份验证的存储库凭证。

有关更多信息，请参阅 在 Amazon ECS 中使用非 AWS 容器映像。

`credentialsParameter`

类型：字符串

必需：是，当使用 `repositoryCredentials` 时

包含私有存储库凭证的密钥的 Amazon 资源名称 (ARN)。

有关更多信息，请参阅 在 Amazon ECS 中使用非 AWS 容器映像。

> **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，使用密钥的完整 ARN 或名称。在您使用 AWS Management Console 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"
        }
    }
]
```

高级容器定义参数

以下高级容器定义参数为 docker run 命令，该命令用于在 Amazon ECS 容器实例上启动容器。

主题

- 运行状况检查
- 环境
- Network settings (网络设置)
- 存储和日志记录
- 安全性
- 资源限制
- Docker 标签

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。有关更多信息，请参阅 使用容器运行状况检查确定 Amazon ECS 任务运行状况。

`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 开头以直接运行命令参数，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，将使用 CMD。

在 AWS Management Console 中注册任务定义时，请使用命令的逗号分隔列表。在创建任务定义后，这些命令将转换为字符串。以下是运行状况检查的输入示例。

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

在使用 AWS Management Console JSON 面板、AWS CLI 或 API 注册任务定义时，将命令列表包含在括号中。以下是运行状况检查的输入示例。

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

不带有 `stderr` 输出的退出代码 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Create a container（创建容器）部分中的 `HealthCheck`。

`interval`

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 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。`startPeriod` 默认已禁用。

环境

`cpu`

类型：整数

必需：否

Amazon ECS 容器代理为容器预留的 `cpu` 单位的数量。在 Linux 上，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。

对于使用 Fargate 启动类型的任务，此字段为可选字段。为任务中的所有容器预留的 CPU 总量必须低于任务级的 cpu 值。

> **Note**
>
> 您可以确定每个 Amazon EC2 实例类型可用的 CPU 单位数。为此，请将 Amazon EC2 实例详细信息页面上为该实例类型列出的 vCPU 数量乘以 1024。

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，假设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。此外，在容器实例上，该任务是运行的唯一任务。在此示例中，容器可以在任何给定时间使用完整的 1024 个 CPU 单位份额。但是，假设您在该容器实例上启动了同一任务的另一个副本。在需要时，保证每个任务至少有 512 个 CPU 单位。同样，如果另一个容器没有使用剩余的 CPU，则每个容器都可以浮动到更高的 CPU 利用率。但是，如果两个任务一直都处于 100% 活动状态，那么它们将限制为 512 个 CPU 单位。

在 Linux 容器实例上，容器实例上的 Docker 进程守护程序使用 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 1024 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 配额将作为绝对配额强制实施。Windows 容器只能访问任务定义中所定义的指定 CPU 量。Null 或零 CPU 值将作为 0 传递给 Docker。然后 Windows 将此值解释为一个 CPU 的 1%。

有关更多示例，请参阅 Amazon ECS 如何管理 CPU 和内存资源。

gpu

类型：ResourceRequirement 对象

必需：否

Amazon ECS 容器代理为容器预留的物理 GPUs 的数量。为某个任务中的所有容器预留的 GPU 的数量不得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 适用于 GPU 工作负载的 Amazon ECS 任务定义。

> **Note**
>
> 在 Fargate 上托管的 Windows 容器不支持此参数。

`Elastic Inference accelerator`

类型：ResourceRequirement 对象

必需：否

对于 `InferenceAccelerator` 类型，`value` 与任务定义中指定的 `InferenceAccelerator` 的 `deviceName` 匹配。有关更多信息，请参阅 the section called “Elastic Inference 加速器名称”。

> **Note**
>
> 自 2023 年 4 月 15 日起，AWS 不再允许新客户加入 Amazon Elastic Inference（EI），并将帮助现有客户将其工作负载迁移到价格更低廉且性能更出色的选项。2023 年 4 月 15 日之后，新客户将无法在 Amazon SageMaker、Amazon ECS 或 Amazon EC2 中使用 Amazon EI 加速器启动实例。但是，在过去 30 天内至少使用过一次 Amazon EI 的客户将视为当前客户，可继续使用该服务。

> **Note**
>
> 在 Fargate 上托管的 Windows 容器不支持此参数。

`essential`

类型：布尔值

必需：否

假设将容器的 `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。然后，将停止属于此任务的所有其他容器。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。假设您有一个由多个容器组成的应用程序。然后，将用于相同目的的容器分组成各组件，并将不同的组件分成多个任务定义。有关更多信息，请参阅 设计适用于 Amazon ECS 的应用程序架构。

```
"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，请参阅 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。此参数映射到 Docker Remote API 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，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。如果有多个参数，则确保每个参数都是数组中的分隔字符串。

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

`workingDirectory`

类型：字符串

必需：否

在其中运行命令的容器中的工作目录。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"
```

`environmentFiles`

类型：对象数组

必需：否

包含要传递到容器的环境变量的文件列表。此参数可将 `--env-file` 选项映射到 docker run。

这不适用于 Windows 容器和 Fargate 上的 Windows 容器

您最多可以指定 10 个环境文件。文件必须具有 `.env` 文件扩展名。环境文件中的每一行都包含 `VARIABLE=VALUE` 格式的环境变量。以 # 开头的行被视为注释并会被忽略。有关适当的环境变量文件语法的更多信息，请参阅在文件中声明默认环境变量。

如果在容器定义中指定了单个环境变量，则这些变量的优先级高于环境文件中包含的变量。如果指定了多个包含相同变量的环境文件，则会按从上到下的顺序处理这些文件。建议您使用唯一的变量名。有关更多信息，请参阅 将单个环境变量传递给 Amazon ECS 容器。

`value`

类型：字符串

必需：是

包含环境变量文件的 Amazon S3 对象的 Amazon Resource Name (ARN)。

`type`

类型：字符串

必需：是

要使用的文件类型 `s3` 是唯一受支持的值。

`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

表示用于对容器开放的密钥的对象。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。

`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

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

> **Note**
>
> 如果 Systems Manager Parameter Store 参数或 Secrets Manager 书与要启动的任务位于同一 AWS 区域中，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"
    }
]
```

Network settings (网络设置)

`disableNetworking`

类型：布尔值

必需：否

当此参数设置为 true 时，容器内会关闭联网。此参数将映射到 Docker Remote API 的Create a container（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。

默认为 `false`。

```
"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被设置为 `bridge` 时才支持此参数。`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，请参阅 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**
>
> 在同一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。容器实例上的网络隔离由安全组和 VPC 设置控制。

```
"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。

默认为 `false`。

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

`mountPoints`

类型：对象数组

必需：否

容器中数据卷的挂载点。此参数将映射到 Docker Remote API 的创建容器部分中的 `Volumes` 以及 docker run 的 `--volume` 选项。

Windows 容器可在 `$env:ProgramData` 所在的驱动器上挂载整个目录。Windows 容器无法在其他驱动器上挂载目录，并且挂载点不能跨驱动器使用。您必须指定挂载点才能将 Amazon EBS 卷直接附加到 Amazon ECS 任务。

`sourceVolume`

类型：字符串

必需：是，当使用 `mountPoints` 时

要挂载的卷的名称。

`containerPath`

类型：字符串

必需：是，当使用 `mountPoints` 时

挂载卷的容器中的路径。

`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

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。

有关使用日志配置的示例任务定义，请参阅Amazon ECS 任务定义示例。

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，容器使用与 Docker 进程守护程序相同的日志记录驱动程序。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，以此来使用不同于 Docker 进程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，考虑以下事项：

- Amazon ECS 支持提供给 Docker 进程守护程序的一小部分日志记录驱动程序。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 或更高版本。
- 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，然后置放在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。
- 对于使用 Fargate 启动类型的任务，您必须在任务之外安装任何其他软件。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","awsfirelens",
      "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","awsfirelens"

必需：是，当使用 `logConfiguration` 时

要用于容器的日志驱动程序。默认情况下，前面列出的有效值是一些日志驱动程序，Amazon ECS 容器代理可以与它们进行通信。

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

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

有关在任务定义中如何使用 `awslogs` 日志驱动程序以将容器日志发送到 CloudWatch Logs 的更多信息，请参阅将 Amazon ECS 日志发送到 CloudWatch 。

有关使用 `awsfirelens` 日志驱动程序的更多信息，请参阅自定义日志路由。

> **Note**
>
> 如果具有未列出的自定义驱动程序，您可以复制 GitHub 上提供的 Amazon ECS 容器代理项目，并对其进行自定义以与该驱动程序一起使用。我们鼓励您针对要包含的更改提交拉取请求。但是，当前不支持运行此软件的修改后副本。

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

`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的键/值映射。

当您使用 FireLens 将日志路由到 AWS 服务 或 AWS Partner Network 目的地以进行日志存储和分析时，您可以将 `log-driver-buffer-limit` 选项设置为限制内存中缓冲的事件数量，然后再发送到日志路由器容器。它可以帮助解决潜在的日志丢失问题，因为高吞吐量可能会导致 Docker 内部缓冲区的内存耗尽。有关更多信息，请参阅 the section called “配置日志以实现高吞吐量”。

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

`secretOptions`

类型：对象数组

必需：否

表示要传递到日志配置的密文的对象。日志配置中使用的密钥可能包括身份验证令牌、证书或加密密钥。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。

`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"
 }]
}
```

`firelensConfiguration`

类型：FirelensConfiguration 对象

必需：否

容器的 FireLens 配置。这用于为容器日志指定和配置日志路由器。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
            "KeyName": ""
        }
    }
}
```

`options`

类型：字符串到字符串映射

必需：否

在配置日志路由器时使用的选项的键/值映射。此字段是可选的，可用于指定自定义配置文件或向日志事件添加其他元数据，如任务、任务定义、群集和容器实例详细信息。如果指定，则使用的语法为 `"options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-value":"arn:aws:s3:::mybucket/fluent.conf|filepath"}`。有关更多信息，请参阅 Amazon ECS 任务定义示例：将日志路由到 FireLens。

`type`

类型：字符串

必需：是

要使用的日志路由器。有效值为 `fluentd` 或 `fluentbit`。

安全性

有关容器安全的更多信息，请参阅 Amazon ECS Best Practices Guide（《Amazon ECS 最佳实践指南》）中的 Task and container security（任务与容器安全）。

`credentialSpecs`

类型：字符串数组

必需：否

SSM 或 Amazon S3 中的 ARN 列表，指向配置容器进行 Active Directory 身份验证的凭证规范（`CredSpec`）文件。建议您使用此参数，而不是 `dockerSecurityOptions`。ARN 的最大数量为 1。

每个 ARN 有两种格式。

credentialspecdomainless:MyARN

您可以使用 `credentialspecdomainless:MyARN` 在 Secrets Manager 中为 `CredSpec` 提供一个密钥的额外部分。您在密钥中提供域的登录凭证。

在任何容器实例上运行的每个任务均可以加入不同的域。

您无需将容器实例加入域，即可使用此格式。

credentialspec:MyARN

您使用 `credentialspec:MyARN` 为单个域提供 `CredSpec`。

在开始使用此任务定义的任何任务之前，您必须将容器实例加入域。

在这两种格式中，将 `MyARN` 替换为 SSM 或 Amazon S3 中的 ARN。

`credspec` 必须在 Secrets Manager 中为包含用户名、密码和要连接的域的密钥提供 ARN。为了提高安全性，该实例未加入域进行无域身份验证。实例上的其他应用程序无法使用无域凭证。您可以使用此参数在同一个实例上运行任务，即使这些任务需要加入不同的域。有关更多信息，请参阅将 gMSA 用于 Windows 容器和将 gMSA 用于 Linux 容器。

`privileged`

类型：布尔值

必需：否

当该参数为 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）。我们建议不要使用 `privileged` 运行容器。在大多数情况下，您可以使用特定参数来指定所需的确切权限，而不是使用 `privileged`。

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。

默认为 `false`。

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

`user`

类型：字符串

必需：否

要在容器内使用的用户。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。

> **Important**
>
> 运行使用 `host` 网络模式的任务时，请勿使用根用户（UID 0）运行容器。作为安全最佳实践，请始终使用非根用户。

您可以使用以下格式指定 `user`。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

有效值："no-new-privileges" | "apparmor:PROFILE" | "label:*value*" | "credentialspec:*CredentialSpecFilePath*"

必需：否

用于为多个安全系统提供自定义配置的字符串列表。有关有效值的更多信息，请参阅 Docker 运行安全配置。此字段对使用 Fargate 启动类型的任务中的容器无效。

对于 EC2 上的 Linux 任务，此参数可用于引用 SELinux 和 AppArmor 多级安全系统的自定义标签。

对于 EC2 上的任何任务，此参数可用于在为 Active Directory 身份验证配置容器时引用凭证规范文件。有关更多信息，请参阅了解了解如何将 gMSA 用于适用于 Amazon ECS 的 EC2 Windows 容器 和将 gMSA 用于 Amazon ECS 上的 EC2 Linux 容器。

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

```
"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。

资源限制

`ulimits`

类型：对象数组

必需：否

列表 ulimit 值为容器定义。此值将覆盖操作系统的默认资源配额设置。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。

Fargate 上托管的 Amazon ECS 任务使用操作系统设置的默认资源限制值，nofile 资源限制参数除外。nofile 资源限制对容器可以使用的打开文件数量设置限制。在 Fargate 上，默认的 nofile 软限制为 1024，硬限制为 65535。您可以将两个限制的值最高设置为 1048576。有关更多信息，请参阅任务资源限制。

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

> **Note**
>
> Windows 容器不支持此参数。

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "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。

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

```
"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 使用控制台创建 Amazon ECS 任务定义。

主题

- Linux 参数
- 容器依赖项
- 容器超时
- 系统控制
- 交互式

- 伪终端

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。有关默认功能和其他可用功能的更多信息，请参阅《Docker 运行参考》中的运行时系统特权和 Linux 功能。有关这些 Linux 功能的更多信息，请参阅 capabilities(7) Linux 手册页面。

`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。此参数将映射到 Docker Remote API 的创建容器部分中的CapAdd以及 Docker 运行的--cap-add选项。

> **Note**
>
> 在 Fargate 上启动的任务仅支持添加 SYS_PTRACE 内核功能。

add

类型：字符串数组

有效值："SYS_PTRACE"

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。此参数将映射到 Docker Remote API 的Create a container（创建容器）部分中的 CapAdd 以及 docker run 的 --cap-add 选项。

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。此参数将映射到 Docker Remote API 的Create a container（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。此参数将映射到 Docker Remote API 的Create a container（创建容器）部分中的 Devices 以及 docker run 的 --device 选项。

> **Note**
>
> 当您使用 Fargate 启动类型或 Windows 容器时，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

要将主机设备公开到的容器中的路径。

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 运行。

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

`maxSwap`

容器可以使用的交换内存总量（以 MiB 为单位）。该参数会转换为 Docker 运行的`--memory-swap`选项，其中，该值为容器内存和`maxSwap`值之和。

如果指定`maxSwap`值为`0`，则该容器不使用交换。接受的值为`0`或任何正整数。如果省略 `maxSwap` 参数，该容器将为其运行所在的容器实例使用交换配置。必须为要使用的`swappiness`参数设置`maxSwap`值。

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

`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 运行。

> **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

`swappiness`

您可以使用此参数调整容器的内存 swappiness 行为。`swappiness` 的值 `0` 防止交换发生，除非有要求。`swappiness` 的值 `100` 导致页面被频繁交换。接受的值为`0`到`100`之间的整数。如果您未指定值，则将使用默认值 `60`。此外，如果未指定 `maxSwap` 的值，则此参数将被忽略。此参数会将`--memory-swappiness`选项映射到 Docker 运行。

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

如果您在 Amazon Linux 2023 上使用任务，则不支持 `swappiness` 参数。

`tmpfs`

tmpfs 挂载的容器路径、挂载选项和最大大小（以 MiB 为单位）。此参数会将`--tmpfs`选项映射到 Docker 运行。

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`

要挂载 tmpfs 卷的绝对文件路径。

类型：字符串

必需：是

`mountOptions`

tmpfs卷挂载选项列表。

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 卷的最大大小（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，请参阅容器依赖项。

> **Note**
>
> 如果容器不满足依赖性约束或在满足约束之前超时，Amazon ECS 不会将依赖性容器推进到其下一个状态。

对于 Amazon EC2 实例上托管的 Amazon ECS 任务，实例至少需要版本 `1.26.0` 的容器代理以启用容器依赖关系。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。如果您正在使用 Amazon ECS 优化型 Amazon Linux AMI，您的实例将需要不低于 `1.26.0-1` 版本的 `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 经 Amazon ECS 优化的 Linux AMI。

对于 Fargate 上托管的 Amazon ECS 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（Linux）或 `1.0.0`（Windows）。

```
"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。此条件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。无法在基本容器上设置此条件。
- SUCCESS - 此条件与 COMPLETE 相同，但它还要求容器退出并具有 zero 状态。无法在基本容器上设置此条件。
- 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容器运行状况检查。这要求从属容器已在任务定义中配置运行状况检查。仅在任务启动时确认此条件。

容器超时

startTimeout

类型：整数

必需：否

示例值：120

在放弃解析容器的依赖项之前要等待的持续时间（以秒为单位）。

例如，在任务定义中指定两个容器，containerA 需要依赖 containerB 达到 COMPLETE、SUCCESS 或 HEALTHY 状态。如果为 containerB 指定了一个 startTimeout 值并且它未在该时间内达到所需状态，则 containerA 将不会启动。

> **Note**
>
> 如果容器不满足依赖性约束或在满足约束之前超时，Amazon ECS 不会将依赖性容器推进到其下一个状态。

对于 Fargate 上托管的 Amazon ECS 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（Linux）。最大值为 120 秒。

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 Fargate 上托管的 Amazon ECS 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（Linux）。如果未指定该参数，则使用默认值 30 秒。最大值为 120 秒。

对于使用 EC2 启动类型的任务，如果未指定 stopTimeout 参数，则使用为 Amazon ECS 容器代理配置变量 ECS_CONTAINER_STOP_TIMEOUT 设置的值。如果既没有设置 stopTimeout 参数，也没有设置 ECS_CONTAINER_STOP_TIMEOUT 代理配置变量，那么，对于 Linux 容器，使用默认值 30 秒；对于 Windows 容器，使用默认值 30 秒。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。有关如何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。如果您正在使用经 Amazon ECS 优化的 Amazon Linux AMI，您的实例将需要不低于 1.26.0-1 版本的 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 经 Amazon ECS 优化的 Linux AMI。

系统控制

systemControls

类型：SystemControl 对象

必需：否

要在容器中设置的命名空间内核参数的列表。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。例如，您可以配置 net.ipv4.tcp_keepalive_time 设置以保持更长期的连接。

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。执行此操作具有以下缺点：

- 对于使用 awsvpc 网络模式的任务，包括 Fargate，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，则以下条件将适用于您的系统控制。有关更多信息，请参阅 IPC 模式。

- 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 值适用于任务中的所有容器。

> **Note**
>
> Windows 容器不支持此参数。

> **Note**
>
> 如果任务使用平台版本 1.4.0 或更高版本（Linux），则在 AWS Fargate 上托管的任务仅支持此参数。Fargate 上的 Windows 容器不支持此参数。

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

namespace

类型：字符串

必需：否

要为其设置 value 的命名空间内核参数。

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

有效的网络命名空间值：开头为 "net.*" 的 Sysctls

Fargate 支持所有此类值。

value

类型：字符串

必需：否

在 namespace 中指定的命名空间内核参数的值。

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。

默认为 false。

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。

默认为 false。

Elastic Inference 加速器名称

> **Note**
>
> 自 2023 年 4 月 15 日起，AWS 不再允许新客户加入 Amazon Elastic Inference（EI），并将帮助现有客户将其工作负载迁移到价格更低廉且性能更出色的选项。2023 年 4 月 15 日之后，新客户将无法在 Amazon SageMaker、Amazon ECS 或 Amazon EC2 中使用 Amazon EI 加速器启动实例。但是，在过去 30 天内至少使用过一次 Amazon EI 的客户将视为当前客户，可继续使用该服务。

您的任务定义的 Elastic Inference 加速器资源要求。有关更多信息，请参阅《Amazon Elastic Inference 开发人员指南》中的什么是 Amazon Elastic Inference？。

任务定义中允许以下参数：

`deviceName`

类型：字符串

必需：是

Elastic Inference 加速器设备名称。还必须在容器定义中引用 `deviceName`，请参阅 `Elastic Inference accelerator`。

`deviceType`

类型：字符串

必需：是

要使用的 Elastic Inference 加速器。

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。预设情况下，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 定义 Amazon ECS 将哪些容器实例用于任务。

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 创建表达式，以为 Amazon ECS 任务定义容器实例。

`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。

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，则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 经 Amazon ECS 优化的 Linux AMI。

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站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 地址的出站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。

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 进程守护程序，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：

- Amazon EBS 卷 – 为数据密集型容器化工作负载提供经济实惠、持久、高性能的块存储。在运行独立任务，或者创建或更新服务时，您可以为每个 Amazon ECS 任务附加 1 个 Amazon EBS 卷。Fargate 或 Amazon EC2 实例上托管的 Linux 任务支持 Amazon EBS 卷。有关更多信息，请参阅 将 Amazon EBS 卷与 Amazon ECS 结合使用。
- Amazon EFS 卷——提供简单的可扩展和持久的文件存储以供您的 Amazon ECS 任务使用。使用 Amazon EFS 时，存储容量是弹性的。它会随着您添加和删除文件而自动增加和缩减。您的应用程序可在需要时获得所需存储。Fargate 或 Amazon EC2 实例上托管的任务支持 Amazon EFS 卷。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。
- FSx for Windows File Server 卷 — 提供完全托管的 Microsoft Windows 文件服务器。这些文件服务器由 Windows 文件系统提供支持。使用 FSx for Windows File Server 和 Amazon ECS 时，您可以

使用永久、分布式、共享的静态文件存储来配置 Windows 任务。有关更多信息，请参阅 将 FSx for Windows File Server 卷与 Amazon ECS 结合使用。

Fargate 上的 Windows 容器不支持此选项。

- Docker 卷 – 在主机 Amazon EC2 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 Amazon EBS）集成。可使用内置 `local` 卷驱动程序或第三方卷驱动程序。只有在 Amazon EC2 实例上运行任务时，才支持 Docker 卷。Windows 容器仅支持使用 `local` 驱动程序。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。
- 绑定挂载 – 挂载到容器中的主机上的文件或目录。在上运行任务时支持绑定挂载主机卷 AWS Fargate 实例或 Amazon EC2 实例。要使用绑定挂载主机卷，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。

有关更多信息，请参阅 Amazon ECS 任务的存储选项。

容器定义中允许以下参数。

`name`

类型：字符串

必需：否

卷的名称。最多允许 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` 对象的 `sourceVolume` 参数中引用。

`host`

必需：否

`host` 参数用于将绑定挂载的生命周期绑定到主机 Amazon EC2 实例（而不是任务）及其存储位置。如果 `host` 参数为空，则 Docker 进程守护程序将为您的数据卷分配一个主机路径，但不保证数据在与其关联的容器停止运行后将保留。

Windows 容器可在 `$env:ProgramData` 所在的驱动器上挂载整个目录。

> **Note**
>
> `sourcePath` 参数仅在使用 Amazon EC2 实例上托管的任务时才支持。

sourcePath

类型：字符串

必需：否

在使用 host 参数时，指定 sourcePath 可声明提供给容器的主机 Amazon EC2 容器实例上的路径。如果此参数为空，则 Docker 进程守护程序将为您分配一个主机路径。如果 host 参数包含 sourcePath 文件位置，则数据卷将在主机 Amazon EC2 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 Amazon EC2 容器实例上不存在 sourcePath 值，则 Docker 进程守护程序将创建该值。如果该位置不存在，则将导出源路径文件夹的内容。

configuredAtLaunch

类型：布尔值

必需：否

指定卷在启动时是否可配置。如果设置为 true，则可以在运行独立任务或者创建或更新服务时配置卷。如果设置为 true，则无法在任务定义中提供其他卷配置。必须将此参数设置为 true 才能将 Amazon EBS 卷配置为附加到一个任务中。将 configuredAtLaunch 设置为 true 并将卷配置延迟到启动阶段使您可以创建不受卷类型或特定卷设置限制的任务定义。这样做可以让您的任务定义在不同的执行环境中重复使用。有关更多信息，请参阅 Amazon EBS 卷。

dockerVolumeConfiguration

类型：DockerVolumeConfiguration 对象

必需：否

使用 Docker 卷时将指定此参数。只有在 EC2 实例上运行任务时，才支持 Docker 卷。Windows 容器仅支持使用 local 驱动程序。要使用绑定挂载，请改为指定 host。

scope

类型：字符串

有效值：task | shared

必需：否

Docker 卷的范围，可确定其生命周期。当任务开始时，将自动预配置范围限定为 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。

`autoprovision`

类型：布尔值

默认值：`false`

必需：否

如果此值为 `true`，则将创建 Docker 卷（如果此卷不存在）。仅在 `scope` 为 `shared` 时使用此字段。如果 `scope` 为 `task`，则必须省略此参数或将其设置为 `false`。

`driver`

类型：字符串

必需：否

要使用的 Docker 卷驱动程序。由于驱动程序值用于任务放置，因此，该名称必须与 Docker 提供的驱动程序名称匹配。如果已使用 Docker 插件 CLI 创建驱动程序，则使用 `docker plugin ls` 可从容器实例中检索驱动程序名称。如果已使用其他方法安装驱动程序，则使用 Docker 插件发现功能可检索驱动程序名称。有关更多信息，请参阅 Docker 插件发现。此参数映射到 Docker Remote API 的创建卷部分中的 `Driver` 以及 `docker volume create` 的 `--driver` 选项。

`driverOpts`

类型：字符串

必需：否

要传递的 Docker 驱动程序特定的选项的映射。此参数映射到 Docker Remote API 的创建卷部分中的 `DriverOpts` 以及 `docker volume create` 的 `--opt` 选项。

`labels`

类型：字符串

必需：否

要添加到 Docker 卷的自定义元数据。此参数映射到 Docker Remote API 的创建卷部分中的 `Labels` 以及 `docker volume create` 的 `--label` 选项。

`efsVolumeConfiguration`

类型：EFSVolumeConfiguration 对象

必需：否

使用 Amazon EFS 卷时将指定此参数。

`fileSystemId`

类型：字符串

必需：是

要使用的 Amazon EFS 文件系统 ID。

`rootDirectory`

类型：字符串

必需：否

Amazon EFS 文件系统中要作为主机内的根目录挂载的目录。如果忽略此参数，将使用 Amazon EFS 卷的根目录。指定 `/` 与忽略此参数效果相同。

> **Important**
>
> 如果在 `authorizationConfig` 中指定了 EFS 接入点，则必须省略根目录值，或者将其设置为 `/`，以便在 EFS 接入点上强制执行设置的路径。

`transitEncryption`

类型：字符串

有效值：`ENABLED | DISABLED`

必需：否

指定是否对 Amazon ECS 主机和 Amazon EFS 服务器之间传输的 Amazon EFS 数据启用加密。如果使用 Amazon EFS IAM 授权，则必须启用传输加密。如果忽略此参数，将使用默认值`DISABLED`。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的加密传输中的数据。

`transitEncryptionPort`

类型：整数

必需：否

在 Amazon ECS 主机和 Amazon EFS 服务器之间发送加密数据时要使用的端口。如果未指定传输加密端口，任务将使用 Amazon EFS 挂载帮助程序使用的端口选择策略。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 EFS 挂载帮助程序。

`authorizationConfig`

类型：EFSAuthorizationConfiguration 对象

必需：否

Amazon EFS 文件系统的授权配置详细信息。

`accessPointId`

类型：字符串

必需：否

要使用的接入点 ID。如果指定了接入点，则必须省略 `efsVolumeConfiguration` 中的根目录值，或者将其设置为 `/`，以便在 EFS 接入点上强制执行设置的路径。如果使用接入点，则必须在`EFSVolumeConfiguration`中启用传输加密。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 接入点。

`iam`

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

指定在挂载 Amazon EFS 文件系统时是否使用在任务定义中定义的 Amazon ECS 任务 IAM 角色。如果启用，则必须在`EFSVolumeConfiguration`中启用传输加密。如果忽略此参数，将使用默认值`DISABLED`。有关更多信息，请参阅适用于任务的 IAM 角色。

`FSxWindowsFileServerVolumeConfiguration`

类型：FSxWindowsFileServerVolumeConfiguration 对象

必需：是

当您使用适用于 Windows File Server 的 Amazon FSx 文件系统进行任务存储时，指定此参数。

`fileSystemId`

类型：字符串

必需：是

要使用的 FSx for Windows File Server 文件系统 ID。

`rootDirectory`

类型：字符串

必需：是

FSx for Windows File Server 文件系统中要作为主机内的根目录挂载的目录。

`authorizationConfig`

`credentialsParameter`

类型：字符串

必需：是

授权凭据选项。

选项

- AWS Secrets Manager 密钥的 Amazon 资源名称（ARN）。
- AWS Systems Manager 参数的 ARN。

`domain`

类型：字符串

必需：是

由 AWS Directory Service for Microsoft Active Directory（AWS Managed Microsoft AD）目录托管的完全限定域名或自托管 EC2 Active Directory。

标签

在注册任务定义时，可以选择指定应用于任务定义的元数据标签。标签可帮助您对任务定义进行分类和组织。每个标签都由一个键和一个可选值组成。您可以同时定义它们。有关更多信息，请参阅 为 Amazon ECS 资源添加标签。

> **Important**
>
> 请勿在标签中添加个人身份信息或其他机密或敏感信息。标签可供许多 AWS 服务访问，包括计费。标签不适合用于私有或敏感数据。

标签对象中允许以下参数。

`key`

类型：字符串

必需：否

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`

类型：字符串

必需：否

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。

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 使用控制台创建 Amazon ECS 任务定义。

主题

- 临时存储
- IPC 模式
- PID 模式

临时存储

`ephemeralStorage`

类型：EphemeralStorage 对象

必需：否

要为任务分配的临时存储量（以 GB 为单位）。此参数用于扩展可用的临时存储总量，超出原定设置数量，可用于在 AWS Fargate 上托管的任务。有关更多信息，请参阅 the section called “绑定挂载”。

> **Note**
>
> 使用平台版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0` 或更高版本（Windows）在 AWS Fargate 上托管的任务仅支持此参数。

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，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。

如果您设置了为任务中的容器使用 `systemControls` 的带有命名空间的内核参数，则以下内容适用于您的 IPC 资源命名空间。有关更多信息，请参阅 系统控制。

- 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

有效值：`host` | `task`

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`。在适用于 Linux 的 Fargate 容器上，唯一的有效值为 `task`。例如，监控 sidecar 可能需要 `pidMode` 访问有关在同一任务中运行的其他容器的信息。

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 实例共享相同的进程命名空间。

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，则默认值为每个容器的私有命名空间。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，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**
>
> Windows 容器不支持此参数。

> **Note**
>
> 如果任务使用平台版本 `1.4.0` 或更高版本（Linux），则在 AWS Fargate 上托管的任务仅支持此参数。Fargate 上的 Windows 容器不支持此参数。

Amazon ECS 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 Amazon ECS 任务定义参数。

Amazon EC2 启动类型模板

```
{
    "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": ""
            }
        ],
        "environmentFiles": [
            {
                "value": "",
                "type": "s3"
            }
        ],
        "mountPoints": [
            {
                "sourceVolume": "",
                "containerPath": "",
                "readOnly": true
            }
        ],
        "volumesFrom": [
            {
                "sourceContainer": "",
                "readOnly": true
            }
        ],
        "linuxParameters": {
            "capabilities": {
                "add": [
                    ""
                ],
                "drop": [
                    ""
                ]
            },
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "read"
                    ]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
```

```
                "containerPath": "",
                "size": 0,
                "mountOptions": [
                    ""
                ]
            }
        ],
        "maxSwap": 0,
        "swappiness": 0
    },
    "secrets": [
        {
            "name": "",
            "valueFrom": ""
        }
    ],
    "dependsOn": [
        {
            "containerName": "",
            "condition": "COMPLETE"
        }
    ],
    "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": "nofile",
                    "softLimit": 0,
                    "hardLimit": 0
                }
            ],
            "logConfiguration": {
                "logDriver": "splunk",
                "options": {
                    "KeyName": ""
                },
                "secretOptions": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ]
            },
            "healthCheck": {
                "command": [
                    ""
                ],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "InferenceAccelerator"
```

```
                }
            ],
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "KeyName": ""
                }
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {
                "sourcePath": ""
            },
            "configuredAtLaunch": true,
            "dockerVolumeConfiguration": {
                "scope": "shared",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {
                    "KeyName": ""
                },
                "labels": {
                    "KeyName": ""
                }
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "DISABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            },
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "authorizationConfig": {
                    "credentialsParameter": "",
                    "domain": ""
```

```
                }
            }
        }
    ],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": ""
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "pidMode": "task",
    "ipcMode": "task",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ],
    "ephemeralStorage": {
        "sizeInGiB": 0
    },
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
```

```
        "operatingSystemFamily": "WINDOWS_SERVER_20H2_CORE"
    }
}
```

Fargate 启动类型模板

> **Important**
>
> 对于 Fargate 启动类型，您必须包括具有以下值之一的 `operatingSystemFamily` 参数：
>
> - `LINUX`
>
> - `WINDOWS_SERVER_2019_FULL`
> - `WINDOWS_SERVER_2019_CORE`
> - `WINDOWS_SERVER_2022_FULL`
> - `WINDOWS_SERVER_2022_CORE`

```
{
    "family": "",
    "runtimePlatform": {"operatingSystemFamily": ""},
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "awsvpc",
    "platformFamily": "",
    "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": ""
                }
            ],
            "environmentFiles": [
                {
                    "value": "",
                    "type": "s3"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "",
                    "containerPath": "",
                    "readOnly": true
                }
            ],
            "volumesFrom": [
                {
                    "sourceContainer": "",
                    "readOnly": true
                }
            ],
            "linuxParameters": {
                "capabilities": {
                    "add": [""],
                    "drop": [""]
                },
                "devices": [
                    {
                        "hostPath": "",
                        "containerPath": "",
                        "permissions": ["read"]
                    }
                ],
                "initProcessEnabled": true,
                "sharedMemorySize": 0,
                "tmpfs": [
```

```
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [""]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "HEALTHY"
            }
        ],
        "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": "msgqueue",
```

```
                "softLimit": 0,
                "hardLimit": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {"KeyName": ""},
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        },
        "healthCheck": {
            "command": [""],
            "interval": 0,
            "timeout": 0,
            "retries": 0,
            "startPeriod": 0
        },
        "systemControls": [
            {
                "namespace": "",
                "value": ""
            }
        ],
        "resourceRequirements": [
            {
                "value": "",
                "type": "GPU"
            }
        ],
        "firelensConfiguration": {
            "type": "fluentd",
            "options": {"KeyName": ""}
        }
    }
],
"volumes": [
    {
        "name": "",
        "host": {"sourcePath": ""},
        "configuredAtLaunch":true,
```

```
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {"KeyName": ""},
                "labels": {"KeyName": ""}
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": 0,
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            }
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "ephemeralStorage": {"sizeInGiB": 0},
    "pidMode": "task",
    "ipcMode": "none",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
```

```
            "deviceType": ""
        }
    ]
}
```

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
```

Amazon ECS 任务定义示例

您可以复制示例和代码段，以开始创建自己的任务定义。

您可以复制这些示例，然后在使用控制台中的通过 JSON 配置选项时粘贴它们。确保自定义示例，例如使用您的账户 ID。您可以在任务定义 JSON 中包含这些代码段。有关更多信息，请参阅使用控制台创建 Amazon ECS 任务定义 和Amazon ECS 任务定义参数。

有关更多任务定义示例，请参阅 GitHub 上的 AWS 示例任务定义。

主题

- Webserver
- splunk 日志驱动程序
- fluentd 日志驱动程序
- gelf 日志驱动程序
- 外部实例上的工作负载
- Amazon ECR 映像和任务定义 IAM 角色
- 带命令的入口点
- 容器依赖项
- Windows 示例任务定义

Webserver

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

```
{
   "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",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
```

```
}
```

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

```
{
    "containerDefinitions": [
        {
            "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"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

splunk 日志驱动程序

以下代码段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 将敏感数据传递给 Amazon ECS 容器。

```
"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"
}],
```

fluentd 日志驱动程序

以下代码段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 将敏感数据传递给 Amazon ECS 容器。

```
"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
 }]
}],
```

gelf 日志驱动程序

以下代码段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。

```
"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
  }
 ]
}],
```

外部实例上的工作负载

注册 Amazon ECS 任务定义时，请使用 requiresCompatibilities 参数并指定 EXTERNAL 在外部实例上运行 Amazon ECS工作负载时验证任务定义是否兼容。如果您使用控制台注册任务定义，则必须使用 JSON 编辑器。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

> **Important**
>
> 如果您的任务需要任务执行 IAM 角色，请确保在任务定义中指定了该角色。

部署工作负载时，请使用 EXTERNAL 启动类型时创建服务或运行独立任务。

以下是此示例的表定义。

Linux

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "nginx",
  "image": "public.ecr.aws/nginx/nginx:latest",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "nginx"
}
```

Windows

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "windows-container",
  "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019",
  "memory": 256,
  "cpu": 512,
  "essential": true,
```

```
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "windows-container"
}
```

Amazon ECR 映像和任务定义 IAM 角色

以下代码段使用 123456789012.dkr.ecr.us-west-2.amazonaws.com 注册表中带 v1 标签的名为 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 Amazon ECS 任务 IAM 角色。

```
{
    "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"
}
```

带命令的入口点

以下代码段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",
                "example.com"
            ],
            "cpu": 16
        }
    ],
    "family": "example_task_2"
}
```

容器依赖项

此代码段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。

```
{
  "family": "appmesh-gateway",
  "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
  },
  "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": "840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-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"
}
```

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": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "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
        }
```

```
      ],
      "memory": 1024,
      "essential": true
    }
  ],
  "networkMode": "awsvpc",
  "memory": "1024",
  "cpu": "1024"
}
```

Amazon ECS 集群

Amazon ECS 集群是任务或服务的逻辑分组。除了任务和服务之外，集群还包含以下资源：

- 可以是以下各项组合的基础设施容量：
  - AWS Cloud 中的 Amazon EC2 实例
  - AWS Cloud 中的无服务器（AWS Fargate (Fargate)）
  - 本地虚拟机（VM）或服务器
- 您的任务和服务运行所在的网络（VPC 和子网）

  当您将 Amazon EC2 实例用于容量时，子网可以位于可用区、本地区域、Wavelength 区域或 AWS Outposts。
- 可选命名空间

  命名空间用于与 Service Connect 进行服务对服务通信。
- 监控选项

  CloudWatch Container Insights 需要支付额外费用，是一项完全托管的服务。它会自动收集、聚合和汇总 Amazon ECS 指标与日志。

以下是有关 Amazon ECS 集群的一般概念。

- Amazon ECS 会创建默认集群。您可以、创建其他集群以分离您的资源。
- 集群是 AWS 区域 特定的。
- 集群的可能状态如下。

  ACTIVE

  集群已准备好接受任务，如果适用，您可以向集群注册容器实例。

  PROVISIONING (正在预置)

  群集具有与其关联的容量提供程序，并且正在创建容量提供程序所需的资源。

  DEPROVISIONING (正在取消预置)

  群集具有与其关联的容量提供程序，并且正在删除容量提供程序所需的资源。

FAILED

群集具有与其关联的容量提供程序，并且容量提供程序所需的资源无法创建。

INACTIVE (非活跃)

集群已删除。具有 INACTIVE 状态的集群可能会在一段时间内在您的账户中保持可被发现。此行为在将来可能会发生变化，因此请确保不依赖于持续存在的 INACTIVE 集群。

- 集群可能包含托管在 AWS Fargate、Amazon EC2 实例 或外部实例上的任务组合。任务可以作为启动类型或容量提供程序策略在 Fargate 或 EC2 基础设施上运行。如果您使用 EC2 作为启动类型，Amazon ECS 不会跟踪和扩缩 Amazon EC2 Auto Scaling 组的容量。有关启动类型的更多信息，请参阅 Amazon ECS 启动类型。
- 集群可能同时包含自动扩缩组容量提供程序和 Fargate 容量提供程序。容量提供程序策略只能包含自动扩缩组容量提供程序或 Fargate 容量提供程序。
- 对于使用 EC2 启动类型或自动扩缩组容量提供程序，您可以使用不同的实例类型。一个实例一次只能注册到一个集群。
- 您可以通过创建自定义 IAM 策略来限制对集群的访问。有关更多信息，请参阅 Amazon Elastic Container Service 的基于身份的策略示例 中的 Amazon ECS 集群示例 部分。
- 您可以使用 Service Auto Scaling 来扩缩 Fargate 任务。有关更多信息，请参阅 自动扩展 Amazon ECS 服务。
- 您可以为集群配置默认 Service Connect 命名空间。设置默认 Service Connect 命名空间后，可以通过启用 Service Connect 将集群中创建的任何新服务添加为命名空间中的客户端服务。无需其他配置。有关更多信息，请参阅 使用 Service Connect 连接具有短名称的 Amazon ECS 服务。

Fargate 启动类型的 Amazon ECS 集群

Amazon ECS 容量提供程序为集群中的任务管理基础设施的扩缩。每个集群可以有一个或多个容量提供程序和一个可选的容量提供程序策略。容量提供程序策略确定任务在集群的容量提供程序之间的分布方式。运行独立任务或创建服务时，您可以使用集群的默认容量提供程序策略，也可以使用覆盖默认策略的容量提供程序策略。

当您在 AWS Fargate 上运行任务时，不需要创建或管理容量。您只需要将以下任一预定义容量提供程序与集群关联：

- Fargate
- Fargate Spot

在 AWS Fargate 容量提供程序上使用 Amazon ECS 使您能够将 Fargate 和 Fargate Spot 容量用于您的 Amazon ECS 任务。

使用 Fargate Spot，您可以按照与 Fargate 价格相比的折扣价格运行能够容忍中断的 Amazon ECS 任务。Fargate Spot 在备用计算容量上运行任务。当 AWS 需要恢复容量时，您的任务将被中断，并发出两分钟的警告。Fargate Spot 仅支持 1.3.0 版或更高版本的平台上使用 X86_64 架构的 Linux 任务。

当使用 Fargate 和 Fargate Spot 容量提供程序的任务停止时，任务状态更改事件会发送到 Amazon EventBridge。停止原因说明了原因。有关更多信息，请参阅 Amazon EC2 任务状态更改事件。

集群可能同时包含 Fargate 容量提供程序和自动扩缩组容量提供程序。但是，容量提供程序策略只能包含 Fargate 容量提供程序或自动扩缩组容量提供程序，而不能同时包含两者。有关更多信息，请参阅自动扩缩组容量提供程序。

使用容量提供程序时，请考虑以下因素：

- 您必须将容量提供程序与集群关联，然后才能将其与容量提供程序策略关联。
- 您最多可以为容量提供程序策略指定 20 个容量提供程序。
- 使用自动扩缩组容量提供程序的服务无法更新为使用 Fargate 容量提供程序，反之亦然。
- 在容量提供程序策略中，如果未在控制台中为容量提供程序指定 `weight` 值，则使用原定设置值 1。如果使用 API或 AWS CLI，则使用原定设置值 `0`。
- 如果在容量提供程序策略中指定了多个容量提供程序，则至少有一个容量提供程序的权重值必须大于零。任何权重为 0 的容量提供程序都不能用于放置任务。如果您在策略中指定的多个容量提供程序的权重全部为 0，则使用该容量提供程序策略的任何 `RunTask` 或 `CreateService` 操作都将失败。
- 在容量提供程序策略中，只能有一个容量提供程序定义了基准值。如果未指定基准值，则使用默认设置值零。
- 集群可能同时包含自动扩缩组容量提供程序和 Fargate 容量提供程序。但是，容量提供程序策略只能包含自动扩缩组容量提供程序或 Fargate 容量提供程序，而不能同时包含两者。
- 集群可能包含使用两个容量提供程序和两种启动类型的服务和独立任务的组合。服务可以更新为使用容量提供程序策略而不是启动类型。但是在执行此操作时必须强制实施新部署。

Fargate Spot 终止通知

在需求极高的时期，Fargate Spot 容量可能会不可用。此操作可能会导致 Fargate Spot 任务被延迟。发生此情况时，Amazon ECS 服务会重试启动任务，直到所需容量变得可用为止。Fargate 不会将 Spot 容量替换为按需容量。

当使用 Fargate Spot 容量的任务因 Spot 中断而停止时，系统会在任务停止之前发送两分钟的警告。警告作为任务状态更改事件发送到 Amazon EventBridge 并作为 SIGTERM 信号发送到正在运行的任务。如果您在服务中使用 Fargate Spot，则在这种情况下，服务调度器将收到中断信号，并在有容量可用时尝试在 Fargate Spot 上启动额外任务。只有一个任务的服务将被中断，直到容量可用。有关正常关闭的更多信息，请参阅使用 ECS 进行正常关闭。

为了确保容器在任务停止之前正常退出，您可以配置以下内容：

- 可以在任务使用的容器定义中指定 120 秒或更小的 stopTimeout 值。默认 stopTimeout 值为 30 秒。您可以指定较长的 stopTimeout 值，为您在收到任务状态更改事件和强制停止容器的时间点之间留出更多时间。有关更多信息，请参阅 容器超时。
- 必须从容器内接收 SIGTERM 信号才能执行任何清理操作。未能处理此信号将导致任务在配置的 stopTimeout 后接收 SIGKILL 信号，并可能导致数据丢失或损坏。

下面是任务状态更改事件的代码段。此代码段显示停止原因和 Fargate Spot 中断的停止代码。

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "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",
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "STOPPED",
    "lastStatus": "RUNNING",
    "stoppedReason": "Your Spot Task was interrupted.",
    "stopCode": "SpotInterruption",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6fEXAMPLE",
    ...
  }
}
```

以下是用于为 Amazon ECS 任务状态更改事件创建 EventBridge 规则的事件模式。您可以选择在 `detail` 字段中指定集群。这样做意味着您将收到该集群的任务状态更改事件。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建 EventBridge 规则

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "clusterArn": [
            "arn:aws:ecs:us-west-2:111122223333:cluster/default"
        ]
    }
}
```

创建 Fargate 启动类型的 Amazon ECS 集群

您可以使用 Amazon ECS 控制台创建 Amazon ECS 集群。开始之前，请确保您已完成 设置以使用 Amazon ECS 中的步骤，然后分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。Amazon ECS 控制台通过创建 AWS CloudFormation 堆栈来创建 Amazon ECS 集群所需的资源。

该控制台自动将 Fargate 和 Fargate Spot 容量提供程序与集群关联。

除集群外，该控制台还会自动创建以下资源：

- AWS Cloud Map 中与集群名称相同的默认命名空间。命名空间允许您在集群中创建的服务无需额外配置即可连接到命名空间中的其他服务。

  有关更多信息，请参阅 互连 Amazon ECS 服务。

您可以修改以下选项：

- 更改与集群关联的默认命名空间。
- 开启 Container Insights。

  CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。Container Insights 还提供诊断信息（如容器重新启动失败），您可以用该信息查明问题并快速

解决问题。有关更多信息，请参阅 the section called “使用 Container Insights 监控 Amazon ECS 容器”。

- 添加标签以帮助您识别集群。

过程

要创建新集群（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在集群配置下，配置以下内容：
   - 对于集群名称，输入唯一的名称。

     该名称最多可以包含 255 个字母（大小写字母）、数字和连字符。
   - （可选）要使用于 Service Connect 的命名空间与集群名称不同，请在命名空间中输入一个唯一的名称。
6. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。
7. （可选）为了帮助识别您的集群，请展开 Tags（标签），然后配置您的标签。

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。

   [删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。
8. 选择创建。

后续步骤

创建集群后，您可以针对应用程序创建任务定义，然后将其作为独立任务或服务的一部分运行。有关更多信息，请参阅下列内容：

- Amazon ECS 任务定义
- 将应用程序作为 Amazon ECS 任务运行
- 使用控制台创建 Amazon ECS 服务

EC2 启动类型的 Amazon ECS 容量提供程序

当您将 Amazon ECS 实例用于容量时，您可以使用自动扩缩组来管理注册到其集群的 Amazon EC2 实例。Auto Scaling 可帮助确保具有正确数量的 Amazon EC2 实例以处理应用程序负载。

您可以使用托管式扩缩功能来让 Amazon ECS 管理自动扩缩组的横向缩减和扩展操作，您也可以自行管理扩缩操作。有关更多信息，请参阅 通过集群自动扩缩功能自动管理 Amazon ECS 容量。

建议您创建新的空自动扩缩组。如果您使用现有 自动扩缩组，则在自动扩缩组用于创建容量提供程序之前，与已在运行并注册到 Amazon ECS 集群的组关联的任何 Amazon EC2 实例都可能无法正确注册到该容量提供程序。使用容量提供程序策略中的容量提供程序时，这可能会引发问题。使用 `DescribeContainerInstances` 可以确认容器实例是否与容量提供程序关联。

> **Note**
>
> 要创建空的 自动扩缩组，请将所需计数设置为零。创建容量提供程序并将其与集群关联后，可以扩展它。
> 当您使用 Amazon ECS 控制台时，Amazon ECS 将代表您创建一个 Amazon EC2 启动模板和自动扩缩组，作为 AWS CloudFormation 堆栈的一部分。它们的前缀为 `EC2ContainerService-<ClusterName>`。您可以将自动扩缩组用作该集群的容量提供程序。

建议您使用托管式实例耗尽功能，以便在不中断工作负载的情况下平稳终止 Amazon EC2 实例。此功能默认启用。有关更多信息，请参阅 安全停止在 EC2 实例上运行的 Amazon ECS 工作负载

在控制台中使用自动扩缩组容量提供程序时应考虑以下因素：

- 自动扩缩组必须具有大于零的 `MaxSize` 才能横向扩展。
- 自动扩缩组不能有实例权重设置。
- 如果自动扩缩组无法横向扩展以适应运行的任务数，则任务将无法转换到 `PROVISIONING` 状态。
- 请勿修改与由容量提供程序管理的自动扩缩组关联的扩展策略资源。

- 如果在创建容量提供程序时启用了托管式扩展，则可将自动扩缩组所需计数设置为 0。启用托管式扩展后，Amazon ECS 管理自动扩缩组的横向缩减和横向扩展操作。
- 您必须将容量提供程序与集群关联，然后才能将其与容量提供程序策略关联。
- 您最多可以为容量提供程序策略指定 20 个容量提供程序。
- 使用自动扩缩组容量提供程序的服务无法更新为使用 Fargate 容量提供程序，反之亦然。
- 在容量提供程序策略中，如果未在控制台中为容量提供程序指定 `weight` 值，则使用原定设置值 1。如果使用 API或 AWS CLI，则使用原定设置值 0。
- 如果在容量提供程序策略中指定了多个容量提供程序，则至少有一个容量提供程序的权重值必须大于零。任何权重为 0 的容量提供程序都不能用于放置任务。如果您在策略中指定的多个容量提供程序的权重全部为 0，则使用该容量提供程序策略的任何 `RunTask` 或 `CreateService` 操作都将失败。
- 在容量提供程序策略中，只能有一个容量提供程序定义了基准值。如果未指定基准值，则使用默认设置值零。
- 集群可能同时包含自动扩缩组容量提供程序和 Fargate 容量提供程序。但是，容量提供程序策略只能包含自动扩缩组容量提供程序或 Fargate 容量提供程序，而不能同时包含两者。
- 集群可能包含使用两个容量提供程序和两种启动类型的服务和独立任务的组合。服务可以更新为使用容量提供程序策略而不是启动类型。但是在执行此操作时必须强制实施新部署。
- Amazon ECS 支持 Amazon EC2 Auto Scaling 暖池。暖池是一组准备投入使用的预初始化 Amazon EC2 实例。每当您的应用程序需要横向扩展时，Amazon EC2 Auto Scaling 都会使用暖池中的预初始化实例，而不是启动冷实例，这允许运行任何最终初始化过程，然后将实例投入使用。有关更多信息，请参阅 为您的 Amazon ECS Auto Scaling 组配置预初始化的实例。

有关创建 Amazon EC2 Auto Scaling 启动模板的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的启动模板。有关创建 Amazon EC2 Auto Scaling 组的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 自动扩缩组。

Amazon ECS 的 Amazon EC2 容器实例安全注意事项

您应该考虑在威胁模型中使用单个容器实例及其访问权限。例如，单个受影响的任务可能能够在同一实例上利用未受影响任务的 IAM 权限。

建议您使用以下方式来帮助防止此情况：

- 运行任务时请勿使用管理员权限。
- 为任务分配具有最低权限访问权限的任务角色。

  容器代理会自动创建具有唯一凭证 ID 的令牌，该令牌用于访问 Amazon ECS 资源。

- 要防止使用 awsvpc 网络模式的任务运行的容器访问提供给 Amazon EC2 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。
- 使用 Amazon GuardDuty 运行时监控来检测 AWS 环境中集群和容器的威胁。运行时监控使用 GuardDuty 安全代理为单个 Amazon ECS 工作负载增加运行时可见性，例如，文件访问、进程执行和网络连接。有关更多信息，请参阅《GuardDuty 用户指南》中的 GuardDuty 运行时监控。

为 Amazon EC2 启动类型创建 Amazon ECS 集群

您可以使用控制台创建 Amazon ECS 集群。开始之前，请确保您已完成 设置以使用 Amazon ECS 中的步骤，然后分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。Amazon ECS 控制台提供了一种简单的方法，通过创建 AWS CloudFormation 堆栈来创建 Amazon ECS 集群所需的资源。

为了使集群创建过程尽可能简单，控制台对许多选项进行了原定设置选择，我们将在下面介绍这些选项。控制台中的大多数部分还提供了帮助面板，以提供进一步的上下文。

您可以在创建集群时注册 Amazon EC2 实例，也可以在创建集群后向集群注册其他实例。

您可以修改以下默认选项：

- 更改您的实例启动所在的子网
- 更改用于控制到容器实例的流量的安全组
- 更改与集群关联的默认命名空间。

  命名空间允许您在集群中创建的服务无需额外配置即可连接到命名空间中的其他服务。默认命名空间与集群名称相同。有关更多信息，请参阅 互连 Amazon ECS 服务。
- 开启 Container Insights。

  CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。Container Insights 还提供诊断信息（如容器重新启动失败），您可以用该信息查明问题并快速解决问题。有关更多信息，请参阅 the section called “使用 Container Insights 监控 Amazon ECS 容器”。
- 添加标签以帮助您识别集群。

自动扩缩组选项

当您使用 Amazon EC2 实例时，必须指定 自动扩缩组来管理运行任务和服务所在的基础设施。

当您选择创建新的 自动扩缩组时，系统会为以下行为自动配置该组：

- Amazon ECS 管理 自动扩缩组的横向缩减和扩展操作。
- Amazon ECS 将防止包含任务且位于 自动扩缩组中的 Amazon EC2 实例在横向缩减过程中终止。有关更多信息，请参阅 AWS Auto Scaling 用户指南中的实例保护。

您可以配置以下 自动扩缩组属性，这些属性决定组要启动的实例的类型和数量：

- 经 Amazon ECS 优化的 AMI。
- 实例类型。
- 连接到实例时，可证明您的身份的 SSH 密钥对。有关如何创建 SSH 密钥的信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对和 Linux 实例。
- 为 自动扩缩组启动的实例的最小数量。
- 为 自动扩缩组启动的实例数的最大数量。

  为了使组横向扩展，最大值必须大于 0。

Amazon ECS 将代表您创建一个 Amazon EC2 Auto Scaling 启动模板和 自动扩缩组，作为 AWS CloudFormation 堆栈的一部分。您为 AMI、实例类型和 SSH 键对指定的值是启动模板的一部分。这些模板的前缀是 EC2ContainerService-<*ClusterName*>，这使得它们很容易识别。自动扩缩组的前缀是 <*ClusterName*>-ECS-Infra-ECSAutoScalingGroup。

为 自动扩缩组启动的实例使用启动模板。

联网选项

默认情况下，实例会启动到该区域的默认子网中。使用当前与子网关联的安全组，而这些安全组会控制您的容器实例的流量。您可以对实例的子网和安全组进行更改。

您可以选择现有子网。您可以使用现有的安全组，也可以创建新的安全组。创建新的安全组时，您需要指定至少一条入站规则。

入站规则会确定哪些流量可以到达您的容器实例，并包括以下属性：

- 要允许的协议
- 允许的端口范围
- 入站流量（来源）

要允许来自特定地址或 CIDR 块的入站流量，请对带有允许的 CIDR 的来源使用自定义。

要允许来自所有目的地的入站流量，请使用 Anywhere 作为来源。这将自动添加 0.0.0.0/0 IPv4 CIDR 块和 ::/0 IPv6 CIDR 块。

要允许来自本地计算机的入站流量，请使用来源组作为来源。这会自动将您的本地计算机的当前 IP 地址添加为允许的来源。

要创建新集群（Amazon ECS 控制台）

在开始之前，分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在集群配置下，配置以下内容：
   - 对于集群名称，输入唯一的名称。

     该名称最多可以包含 255 个字母（大小写字母）、数字和连字符。
   - （可选）要使用于 Service Connect 的命名空间与集群名称不同，请在命名空间中输入一个唯一的名称。
6. 向集群添加 Amazon EC2 实例，展开基础设施，清除 AWS Fargate（无服务器），然后选择 Amazon EC2 实例。接下来，配置充当容量提供程序的 自动扩缩组：
   a. 要使用现有 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择组。
   b. 要创建 自动扩缩组，请从 自动扩缩组（ASG）（自动扩缩组（ASG））中，选择 Create new group（创建新组），然后提供有关组的以下详细信息：
      - 对于预置模型，选择使用按需型实例还是竞价型实例。
      - 如果您选择使用竞价型实例，则对于分配策略，请选择哪些 Spot 容量池（实例类型和可用区）用于该实例。

        对于大多数工作负载，您可以选择价格容量优化。

        有关更多信息，请参阅《Amazon EC2 用户指南》中的竞价型实例的分配策略。

- 对于Operating system/Architecture（操作系统/架构），为 自动扩缩组实例选择经 Amazon ECS 优化的 AMI。
- 对于 EC2 instance type（EC2 实例类型），选择工作负载的实例类型。

  如果 自动扩缩组使用相同或相似的实例类型，则托管扩展效果最佳。
- 对于 EC2 实例角色，请选择现有的容器实例角色，也可以创建一个新的容器实例角色。

  有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。
- 对于 Capacity（容量），输入 自动扩缩组中启动的实例数的最小值和最大值。
- 对于 SSH key pair（SSH 密钥对），连接到实例时，选择可证明您身份的密钥对。
- 要允许更大的图像和存储空间，请在根 EBS 卷大小中输入以 GiB 为单位的值。

7. （可选）要更改 VPC 和子网，请在 Amazon EC2 实例的联网下，执行以下任一操作：

- 要删除子网，请在 Subnets（子网）下，为您要删除的每个子网选择 X。
- 要更改为默认 VPC 以外的 VPC，在 VPC 下，选择现有的 VPC，然后在子网中，选择子网。
- 选择安全组。在安全组下，选择下列选项之一：
  - 要选择现有安全组，请选择使用现有安全组，然后选择安全组。
  - 要创建安全组，请选择创建新安全组。然后为每条入站规则选择添加规则。

    有关入站规则的信息，请参阅 联网选项。
- 要自动分配公有 IP 地址到您的 Amazon EC2 容器实例，请为自动分配公有 IP 选择以下选项之一：
  - 使用子网设置 – 当实例启动所在的子网是公有子网时，为实例分配公有 IP 地址。
  - 开启 – 为实例分配公有 IP 地址。

8. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。

9. （可选）

   如果您使用带手动选项的运行时监控，并且希望由 GuardDuty 监视此集群，则请选择添加标签并执行以下操作：

- 对于键，输入 **`guardDutyRuntimeMonitoringManaged`**
- 对于值，请输入 **`true`**。

10. （可选）要管理集群标签，请展开 Tags（标签），然后执行以下操作之一：

[添加标签] 选择 Add tag（添加标签），然后执行以下操作：

- 对于 Key（键），输入键名称。
- 对于值，输入键值。

[删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。

11. 选择创建。

后续步骤

创建集群后，您可以针对应用程序创建任务定义，然后将其作为独立任务或服务的一部分运行。有关更多信息，请参阅下列内容：

- Amazon ECS 任务定义
- 将应用程序作为 Amazon ECS 任务运行
- 使用控制台创建 Amazon ECS 服务

通过集群自动扩缩功能自动管理 Amazon ECS 容量

Amazon ECS 可以管理注册到您的集群的 Amazon EC2 实例的扩展。这称为 Amazon ECS 集群自动扩缩。当创建 Amazon ECS Auto Scaling 组容量提供程序时，您可以开启托管式扩缩。然后，您可以为此自动扩缩组中的实例的利用率设置目标百分比（`targetCapacity`）。Amazon ECS 会创建两个自定义 CloudWatch 指标和自动扩缩组的目标跟踪扩缩策略。然后，Amazon ECS 根据您的任务使用的资源利用率来管理横向缩减和横向扩展操作。

对于与集群关联的每个自动扩缩组容量提供程序，Amazon ECS 将创建和管理以下资源：

- 指标值低 CloudWatch 告警
- 指标值高 CloudWatch 告警
- 目标跟踪扩展策略

> **Note**
>
> Amazon ECS 创建目标跟踪扩展策略并将其附加到自动扩缩组。要更新目标跟踪扩展策略，应更新容量提供程序管理的扩展设置，而不是直接更新扩展策略。

当您关闭托管式扩展或取消容量提供程序与集群的关联时，Amazon ECS 将删除 CloudWatch 指标以及目标跟踪扩展策略资源。

Amazon ECS 使用以下指标确定要采取的操作：

`CapacityProviderReservation`

用于特定容量提供程序的容器实例的百分比。Amazon ECS 会生成此指标。

Amazon ECS 将 `CapacityProviderReservation` 值设置为 0-100 之间的数字。Amazon ECS 使用以下公式来表示自动扩缩组中剩余容量的比率。然后，Amazon ECS 将指标发布给 CloudWatch。有关计算指标的方式的更多信息，请参阅深入了解 Amazon ECS 集群自动扩缩。

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

`DesiredCapacity`

自动扩缩组的容量值。此指标尚未发布到 CloudWatch。

Amazon ECS 将 `CapacityProviderReservation` 指标发布给 `AWS/ECS/ManagedScaling` 命名空间中的 CloudWatch。`CapacityProviderReservation` 指标会导致以下操作之一：

`CapacityProviderReservation` 值等于 `targetCapacity`

自动扩缩组不需要横向缩减或横向扩展。已达到目标利用率的百分比。

`CapacityProviderReservation` 值大于 `targetCapacity`

有更多的任务使用的容量百分比高于您的 `targetCapacity` 百分比。`CapacityProviderReservation` 指标值的增加会导致关联的 CloudWatch 警报生效。此告警会更新 自动扩缩组的 `DesiredCapacity` 值。自动扩缩组使用此值启动 EC2 实例，然后向集群注册它们。

当 `targetCapacity` 为默认值 100% 时，新任务将在横向扩展期间处于 `PENDING` 状态，因为实例上没有可用容量来运行这些任务。在新实例向 ECS 进行注册后，这些任务将在新实例上启动。

`CapacityProviderReservation` 值小于 `targetCapacity`

使用比您的 `targetCapacity` 百分比低的容量百分比的任务较少，并且至少有一个实例可以被终止。`CapacityProviderReservation` 指标值的减少会导致关联的 CloudWatch 警报生效。此告警会更新 自动扩缩组的 `DesiredCapacity` 值。自动扩缩组使用此值终止 EC2 容器实例，然后从集群中将其注销。

自动扩缩组遵照组终止策略来确定它在横向缩减事件期间首先终止哪些实例。此外，它还可以避免启用实例横向缩减保护设置的实例。如果您开启托管终止保护，集群自动扩缩可以管理哪些实例具有实例横向缩减保护设置。有关托管终止保护的更多信息，请参阅 控制 Amazon ECS 终止的实例。有关自动扩缩组终止实例方式的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的控制在横向缩减期间终止哪些自动扩缩实例。

使用集群自动扩缩时应考虑以下因素：

- 请勿更改或管理与容量提供商关联的自动扩缩组的所需容量，该组具有除 Amazon ECS 管理的扩展策略之外的任何扩展策略。
- Amazon ECS 使用 `AWSServiceRoleForECS` 服务相关 IAM 角色来获取它代表您调用 AWS Auto Scaling 所需的权限。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。
- 将容量提供程序与自动扩缩组结合使用时，对于创建容量提供程序的用户、组或角色需要 `autoscaling:CreateOrUpdateTags` 权限。这是因为Amazon ECS在将 自动扩缩组与容量提供程序关联时会向其添加一个标签。

> **Important**
>
> 确保您使用的任何工具都不会从自动扩缩组中删除 `AmazonECSManaged` 标签。如果删除此标签，则 Amazon ECS 无法管理扩缩。

- 集群自动扩缩不会修改该组的 MinimumCapacity 或 MaximumCapacity。为了使组横向扩展，MaximumCapacity 的值必须大于 0。
- 启用 Auto Scaling（托管式扩展）时，容量提供程序只能同时连接到一个集群。如果您的容量提供程序已关闭托管扩缩，则可以将其与多个集群关联。
- 关闭托管式扩展后，容量提供程序不会横向缩减或横向扩展。您可以使用容量提供程序策略来平衡容量提供程序之间的任务。
- 就容量而言，`binpack` 策略是最有效的策略。
- 当目标容量低于 100% 时，放置策略需要使用 `binpack` 策略，而不是使用比 `binpack` 策略更高阶的 `spread` 策略。这样可以防止容量提供程序在每个任务都有专用实例或达到限制之前横向扩展。

优化 Amazon ECS 集群自动扩缩

在 Amazon EC2 上运行 Amazon ECS 的客户可以利用集群自动扩缩来管理 Amazon EC2 Auto Scaling 组的扩缩。通过集群自动扩缩，您可以将 Amazon ECS 配置为自动扩缩您的自动扩缩组，您

只需专注于运行任务即可。Amazon ECS 将确保自动扩缩组根据需要横向缩减和横向扩展，无需进一步干预。Amazon ECS 容量提供程序用于通过确保有足够的容器实例可以满足应用程序的需求来管理集群中的基础设施。要了解集群自动扩缩在后台的工作方式，请参阅深入了解 Amazon ECS 集群自动扩缩。

集群自动扩缩依赖于基于 CloudWatch 与自动扩缩组的集成来调整集群容量。因此，它具有与发布 CloudWatch 指标相关的固有延迟、指标 `CapacityProviderReservation` 突破 CloudWatch 警报（包括高警和低警报）所花费的时间以及新启动的 Amazon EC2 实例预热所花费的时间。您可以执行以下操作来提高集群自动扩缩的响应能力，从而加快部署速度：

容量提供程序分步扩缩大小

Amazon ECS 容量提供程序最终将增加/缩小容器实例，以满足您的应用程序需求。Amazon ECS 将启动的实例最小数量默认设置为 1。如果需要多个实例来放置待处理的任务，这可能会额外增加您的部署时间。您可以通过 Amazon ECS API 增加 `minimumScalingStepSize`，以增加 Amazon ECS 一次横向缩减或扩展的最小实例数量。过低的 `maximumScalingStepSize` 可能会限制一次横向缩减或扩展的容器实例数量，这可能会减慢您的部署速度。

> **Note**
>
> 此配置目前只能通过 `CreateCapacityProvider` 或 `UpdateCapacityProvider` API 进行。

实例预热期

实例预热期是新启动的 Amazon EC2 实例可以向自动扩缩组提供 CloudWatch 指标之后需要经过的时间。指定的预热期到期后，该实例将计入自动扩缩组的聚合指标，集群自动扩缩将继续进行下一次计算迭代，以估计所需的实例数量。

`instanceWarmupPeriod` 的默认值为 300 秒，您可以通过 `CreateCapacityProvider` 或 `UpdateCapacityProvider` API 将其配置为较低的值，以提高缩放响应速度。

备用容量

如果您的容量提供程序没有可用于放置任务的容器实例，则它需要通过即时启动 Amazon EC2 实例来增加（横向扩展）集群容量，并等待它们启动后才能在其上启动容器。这样会大大降低任务启动率。您在此有两种选择。

在这种情况下，已经启动并准备好运行任务的备用 Amazon EC2 容量将提高有效的任务启动率。您可以使用 `Target Capacity` 配置来表示您希望在集群中维护备用容量。例如，通过将 `Target`

Capacity 设置为 80%，表示您的集群始终需要 20% 的备用容量。此备用容量可使任何独立任务立即启动，从而确保任务启动不受限制。这种方法的代价是保持备用集群容量的潜在成本增加。

您可以考虑采用另一种方法，即为服务而不是容量提供程序增加余量。这意味着，与其减少 Target Capacity 配置以启动备用容量，不如修改服务自动扩缩的目标跟踪扩缩指标或分步扩缩阈值来增加服务中的副本数量。请注意，这种方法仅对高峰工作负载有帮助，但在部署新服务并第一次从 0 个任务增加到 N 个任务时不会产生任何影响。有关相关扩缩策略的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的目标跟踪扩缩策略或分步扩缩策略。

控制 Amazon ECS 终止的实例

> **Important**
>
> 您必须在自动扩缩组上启用自动扩缩实例横向缩减保护才能使用集群自动扩缩的托管终止保护功能。

通过托管式终止保护，集群自动扩缩可以控制终止哪些实例。当您使用托管式终止保护时，Amazon ECS 仅终止没有任何正在运行的 Amazon ECS 任务的 EC2 实例。由使用 DAEMON 计划策略的服务运行的任务将被忽略，即使实例正在运行这些任务，也可以通过集群自动扩缩终止该实例。这是因为集群中所有实例均在运行这些任务。

Amazon ECS 首先为自动扩缩组中的 EC2 实例开启实例横向缩减保护选项。然后，Amazon ECS 将任务放在实例上。当实例上的所有非守护进程任务停止时，Amazon ECS 将启动横向缩减过程并关闭 EC2 实例的横向缩减保护。然后，自动扩缩组可以终止实例。

自动扩缩实例横向缩减保护控制自动扩缩可以终止哪些 EC2 实例。在横向缩减过程中，开启了横向缩减功能的实例无法终止。有关自动扩缩实例横向缩减保护的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的使用实例横向缩减保护。

您可以设置 targetCapacity 百分比，以便有备用容量。这有助于更快地启动未来的任务，因为自动扩缩组不必启动更多实例。Amazon ECS 使用目标容量值来管理该服务创建的 CloudWatch 指标。Amazon ECS 管理 CloudWatch 指标。自动扩缩组被视为稳定状态，因此不需要执行扩缩操作。这些值可以是 0-100%。例如，要将 Amazon ECS 配置为在 Amazon ECS 任务使用的容量之外保持 10% 的可用容量，请将目标容量值设置为 90%。在容量提供程序上设置 targetCapacity 值时，请考虑以下因素。

- 小于 100% 的 targetCapacity 值表示集群中需要存在的可用容量（Amazon EC2 实例）的量。可用容量意味着没有正在运行的任务。

- 可用区之类的放置约束无需额外的 binpack 即可强制 Amazon ECS 最终为每个实例运行一个任务，这可能不是所需的行为。

您必须在自动扩缩组上启用自动扩缩实例横向缩减保护才能使用托管终止保护功能。如果您不开启横向缩减保护，则开启托管终止保护可能会导致不良行为。例如，您的实例可能停留在耗尽状态。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南 中的使用实例横向缩减保护。

当您对容量提供程序使用终止保护时，请勿在与容量提供程序关联的自动扩缩组上执行任何手动操作，例如分离实例。手动操作可能会中断容量提供程序的横向缩减操作。如果您将某个实例与自动扩缩组分离，则还需要从 Amazon ECS 集群中取消注册已分离的实例。

托管扩展行为

当您拥有使用托管扩缩的自动扩缩组容量提供程序时，Amazon ECS 估计要添加到集群的最佳实例数，并使用该值确定要请求的实例数。

Amazon ECS 通过遵循服务、独立任务或集群默认设置中的容量提供程序策略，为每项任务选择容量提供程序。对于单个容量提供程序，Amazon ECS 将按照其余步骤进行操作。

没有容量提供程序策略的任务会被容量提供程序忽略。没有容量提供程序策略的待处理任务不会导致任何容量提供程序横向扩展。如果任务或服务设置了启动类型，则该任务或服务无法设置容量提供程序策略。

下面更详细地描述了扩展行为。

- 对此容量提供程序的所有预调配任务进行分组，以便每个组具有相同的精确资源需求。
- 在自动扩缩组中使用多个实例类型时，自动扩缩组中的实例类型将按其参数进行排序。这些参数包括 vCPU、内存、弹性网络接口 (ENI)、端口和 GPU。选择每个参数的最小和最大实例类型。有关如何选择实例类型的更多信息，请参阅 Amazon ECS 的 Amazon EC2 容器实例。

> **Important**
>
> 如果一组任务的资源需求高于自动扩缩组中最小的实例类型，则该组任务无法使用此容量提供程序运行。容量提供程序不对自动扩缩组进行扩展。任务仍处于 PROVISIONING 状态。为防止任务保持 PROVISIONING 状态，建议您针对不同的最低资源要求创建单独的自动扩缩组和容量提供程序。运行任务或创建服务时，仅将容量提供程序添加到容量提供程序策略中，该策略可以在自动扩缩组中最小的实例类型上运行任务。对于其他参数，您可以使用放置约束

- 对于每组任务，Amazon ECS 计算运行未放置任务所需的实例数。此计算使用 binpack 策略。此策略考虑任务的 vCPU、内存、弹性网络接口 (ENI)、端口和 GPU 要求。它还考虑 Amazon EC2 实例的资源可用性。将最大实例类型的值视为最大计算实例计数。最小实例类型的值用作保护。如果最小的实例类型无法运行任务的至少一个实例，则计算会将该任务视为不兼容。因此，该任务会被排除在横向扩展计算之外。当所有任务都与最小的实例类型不兼容时，集群自动扩缩将停止，CapacityProviderReservation 值保持为 targetCapacity 值。
- 如果出现以下任一情况，Amazon ECS 会将与 CloudWatch 相关的 CapacityProviderReservation 指标发布给 minimumScalingStepSize。
  - 计算的最大实例计数小于最小扩缩步长大小。
  - 小于 maximumScalingStepSize 或最大计算的实例计数的较低值。
- CloudWatch 警报使用容量提供程序的 CapacityProviderReservation 指标。当 CapacityProviderReservation 指标值大于 targetCapacity 值时，告警也会增加自动扩缩组的 DesiredCapacity。targetCapacity 值是在集群自动扩缩激活期间发送到 CloudWatch 警报的容量提供程序设置。

  默认 targetCapacity 为 100%。
- 自动扩缩组启动其他 EC2 实例。为防止过度预置，自动扩缩确保最近启动的 EC2 实例容量在启动新实例之前稳定下来。Auto Scaling 会检查是否所有现有实例都已超过 instanceWarmupPeriod（现在减去实例启动时间）。instanceWarmupPeriod 中的实例无法横向扩展。

  新启动实例的默认预热时间为 300 秒。

有关更多信息，请参阅深入了解 Amazon ECS 集群自动扩缩。

横向扩展注意事项

对于横向扩展过程，请考虑以下几点：

- 尽管存在多个放置约束，但我们建议您只使用 distinctInstance 任务放置约束。这可以防止横向扩展过程停止，因为您使用的是与采样实例不兼容的放置约束。
- 如果 自动扩缩组使用相同或相似的实例类型，则托管扩展效果最佳。
- 当需要横向扩展过程且当前没有正在运行的容器实例时，Amazon ECS 最初总是横向扩展到两个实例，然后执行额外的横向扩展或横向缩减过程。任何额外的横向扩展均会等待实例预热期。对于横向缩减过程，Amazon ECS 在横向扩展过程结束后会等待 15 分钟，然后才会一直启动横向缩减过程。

- 第二个横向扩展步骤需要等到 `instanceWarmupPeriod` 到期，这可能会影响整体扩展限制。如果您需要缩短此时间，请确保 `instanceWarmupPeriod` 足够大，以便 EC2 实例启动和开始 Amazon ECS 代理（这可防止过度预置）。
- 集群自动扩缩支持容量提供程序自动扩缩组中的启动配置、启动模板和多种实例类型。您还可以使用基于属性的实例类型选择而不使用多个实例类型。
- 使用带有按需实例和多个实例类型或竞价型实例的 自动扩缩组时，请将较大的实例类型放在优先级列表中，并且不指定权重。这时不支持指定权重。有关详细信息，请参见 AWS Auto Scaling 用户手册中的使用多个实例类型 自动扩缩组。
- 然后，Amazon ECS 将启动 `minimumScalingStepSize`，如果计算的最大实例计数小于最小扩展步长大小，或者 `maximumScalingStepSize` 或最大计算的实例计数值。
- 如果 Amazon ECS 服务或 `run-task` 启动了一个任务且容量提供程序容器实例没有足够的资源来开始任务，则 Amazon ECS 会限制每个集群具有此状态的任务数量，并防止任何任务超过此限制。有关更多信息，请参阅 服务限额。

托管的横向缩减行为

Amazon ECS 监控集群内每个容量提供程序的容器实例。当容器实例没有运行任何任务时，容器实例被视为空且 Amazon ECS 将启动横向缩减过程。

CloudWatch 横向缩减告警需要 15 个数据点（15 分钟），然后才能开始 自动扩缩组的横向缩减过程。在横向缩减过程开始之后，直到 Amazon ECS 需要减少已注册容器实例的数量时，自动扩缩组将 `DesireCapacity` 值设置为每分钟大于一个实例和少于 50%。

当 Amazon ECS 在横向缩减过程中请求横向扩展时（`CapacityProviderReservation` 大于 100 时），横向缩减过程将停止，如果需要，将从头开始。

下面更详细地描述了横向缩减行为：

1. Amazon ECS 计算空容器实例的数量。当没有运行进程守护程序任务时，容器实例被视为空。
2. Amazon ECS 将 `CapacityProviderReservation` 值设置为 0-100 之间的数字，该数字使用以下公式表示自动扩缩组需要的大小与实际大小的比值，以百分比表示。然后，Amazon ECS 将指标发布给 CloudWatch。有关计算指标的方式的更多信息，请参阅深入了解 Amazon ECS 集群自动扩缩

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

3. CapacityProviderReservation 指标会生成 CloudWatch 告警。此告警会更新 自动扩缩组的 DesiredCapacity 值。然后，发生下列任一操作时：
   - 如果您不使用容量提供程序托管式终止，自动扩缩组将使用自动扩缩组终止策略选择 EC2 实例，并终止这些实例，直至 EC2 实例的数量达到 DesiredCapacity。然后，容器实例将从集群中注销。
   - 如果所有容器实例都使用托管终止保护，Amazon ECS 将删除对空容器实例的横向缩减保护。然后，自动扩缩组将能够终止 EC2 实例。然后，容器实例将从集群中注销。

开启 Amazon ECS 集群自动扩缩

您可以使用 AWS CLI 开启集群自动扩缩。

在开始前，请创建自动扩缩组和容量提供程序。有关更多信息，请参阅 the section called “EC2 启动类型的容量提供程序”。

要开启集群自动扩缩，您需要将容量提供程序与集群相关联，然后开启集群自动扩缩。

1. 使用 put-cluster-capacity-providers 命令以将一个或多个容量提供程序与集群关联。

   要添加 AWS Fargate 容量提供程序，请在请求中包含 FARGATE 和 FARGATE_SPOT 容量提供程序。有关更多信息，请参阅 AWS CLI 命令参考 中的 put-cluster-capacity-providers。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

   要为 EC2 启动类型添加自动扩缩组，请在请求中包含自动扩缩组名称。有关更多信息，请参阅 AWS CLI 命令参考 中的 put-cluster-capacity-providers。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

2. 使用 describe-clusters 命令以验证关联是否成功。有关更多信息，请参阅 AWS CLI 命令参考 中的 describe-clusters。

   ```
   aws ecs describe-clusters \
   ```

```
  --cluster ClusterName \
  --include ATTACHMENTS
```

3. 使用 update-capacity-provider 命令以为容量提供程序开启托管自动扩缩。有关更多信息，请参阅 AWS CLI 命令参考 中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=ENABLED
```

关闭 Amazon ECS 集群自动扩缩

您可以使用 AWS CLI 关闭集群自动扩缩。

要关闭集群的集群自动扩缩，您可以在集群中取消容量提供程序与启用的托管式扩缩的关联，也可以更新容量提供程序以关闭托管式扩缩。

取消容量提供程序的关联

按照以下步骤将容量提供程序与取消集群关联。

1. 使用 put-cluster-capacity-providers 命令来取消自动扩缩组容量提供程序与集群的关联。集群可以与 AWS Fargate 容量提供程序保持关联。有关更多信息，请参阅 AWS CLI 命令参考 中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers FARGATE FARGATE_SPOT \
  --default-capacity-provider-strategy '[]'
```

使用 put-cluster-capacity-providers 命令来取消自动扩缩组容量提供程序与集群的关联。有关更多信息，请参阅 AWS CLI 命令参考 中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers [] \
  --default-capacity-provider-strategy '[]'
```

2. 使用 describe-clusters 命令以验证取消关联是否成功。有关更多信息，请参阅 AWS CLI 命令参考 中的 describe-clusters。

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

为容量提供程序关闭托管扩缩

按照以下步骤关闭容量提供程序的托管扩缩。

- 使用 update-capacity-provider 命令以为容量提供程序关闭托管自动扩缩。有关更多信息，请参阅 AWS CLI 命令参考 中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=DISABLED
```

安全停止在 EC2 实例上运行的 Amazon ECS 工作负载

托管实例耗尽有助于优雅终止 Amazon EC2 实例。这使您的工作负载可以安全停止并重新安排到非终止实例。执行基础设施维护和更新时无需担心工作负载中断。通过使用托管实例耗尽功能，您可以简化需要更换 Amazon EC2 实例的基础设施管理工作流程，同时确保应用程序的恢复能力和可用性。

Amazon ECS 托管实例耗尽与自动扩缩组实例替换配合使用。根据实例刷新和最长实例生命周期，客户可以确保其容量符合最新的操作系统和安全规定。

托管式实例耗尽功能只能与 Amazon ECS 容量提供程序结合使用。您可以在使用 Amazon ECS、AWS CLI 或 SDK 创建或更新自动扩缩组容量提供程序时，开启托管式实例耗尽功能。

Amazon ECS 托管实例耗尽涵盖了以下事件。

- 自动扩缩组实例刷新 – 使用实例刷新来滚动替换自动扩缩组中的 Amazon EC2 实例，而不是手动批量替换。当您需要替换大量实例时，这将有用。实例刷新是通过 Amazon EC2 控制台或 StartInstanceRefresh API 启动的。如果您使用托管终止保护，则请确保在调用 StartInstanceRefresh 时为横向缩减保护选择 Replace。
- 最长实例生命周期 – 在替换自动扩缩组实例时，您可以定义最长生命周期。这有助于根据内部安全策略或合规性安排替换实例。

- 自动扩缩组横向缩减 – 根据扩缩策略和计划的扩缩操作，自动扩缩组支持自动扩缩实例。通过将自动扩缩组用作 Amazon ECS 容量提供程序，您可以在自动扩缩组实例中没有任务运行时横向缩减这些实例。
- 自动扩缩组运行状况检查 –自动扩缩组支持许多运行状况检查，以管理运行不正常的实例的终止。
- AWS CloudFormation 堆栈更新 – 您可以向 AWS CloudFormation 堆栈添加 `UpdatePolicy` 属性，以便在组更改时执行滚动更新。
- 竞价型容量再平衡 – 自动扩缩组尝试根据 Amazon EC2 容量再平衡通知主动替换中断风险较高的竞价型实例。替换实例启动且正常运行后，自动扩缩组会终止旧实例。Amazon ECS 托管式实例耗尽功能会像耗尽非竞价型实例一样耗尽竞价型实例。
- Spot 中断 – 竞价型实例将在通知两分钟后终止。Amazon ECS 托管实例耗尽会将实例置于耗尽状态作为响应。

Amazon EC2 Auto Scaling 生命周期挂钩（托管实例耗尽）

自动扩缩组生命周期挂钩使客户能够创建由实例生命周期中的特定事件触发的解决方案，并在该特定事件发生时执行自定义操作。一个自动扩缩组最多允许 50 个挂钩。可存在多个终止挂钩并可并行执行，自动扩缩组会等待所有挂钩完成后再终止实例。

除了 Amazon ECS 托管挂钩终止外，您还可以配置自己的生命周期终止挂钩。生命周期挂钩有 `default action`，建议将 `continue` 设置为默认操作，以确保其他挂钩（例如 Amazon ECS 托管式挂钩）不会受到自定义挂钩的任何错误的影响。

如果您已经配置了自动扩缩组终止生命周期挂钩并启用了 Amazon ECS 托管式实例耗尽功能，则两个生命周期挂钩都将执行。但不能保证相对时间。生命周期挂钩有一个 `default action` 设置，用于指定超时过后要采取的操作。如果失败，则建议使用 `continue` 作为自定义挂钩中的默认结果。这样可以确保其他挂钩（尤其是 Amazon ECS 托管式挂钩）不会受到自定义生命周期挂钩中任何错误的影响。`abandon` 的替代结果会导致跳过所有其他挂钩，因此应避免使用。有关自动扩缩组生命周期挂钩的更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 Amazon EC2 Auto Scaling 生命周期挂钩。

任务和托管实例耗尽

Amazon ECS 托管实例耗尽使用容器实例中现有的耗尽功能。容器实例耗尽功能对属于 Amazon ECS 服务的副本任务执行替换和停止。处于 `PENDING` 或 `RUNNING` 状态的独立任务（如由 `RunTask` 调用的任务）将不受影响。您必须等待这些任务完成或手动停止这些任务。容器实例将保持 `DRAINING` 状态，直到所有任务均已停止或经过 48 小时。进程守护程序任务是在所有副本任务停止后最后停止的任务。

托管实例耗尽和托管终止保护

即使禁用了托管终止，托管式实例耗尽功能也会起作用。有关托管式终止保护的信息，请参阅 控制 Amazon ECS 终止的实例。

下表总结了托管终止和托管耗尽不同组合的行为。

托管终止	托管耗尽	结果
已启用	已启用	Amazon ECS 可保护正在运行任务的 Amazon EC2 实例不被横向缩减事件终止。任何正在终止的实例（例如未设置终止保护、已收到 Spot 中断或因实例刷新而强制终止的实例）都将正常地耗尽。
已禁用	已启用	Amazon ECS 无法保护运行任务的 Amazon EC2 实例免于横向缩减。但是，任何正在终

托管终止	托管耗尽	结果
		止的实例都会正常地耗尽。
已启用	已禁用	Amazon ECS 可保护正在运行任务的 Amazon EC2 实例不被横向缩减事件终止。但是，实例仍可能因 Spot 中断、强制实例刷新或其未运行任何任务而终止。Amazon ECS 不会对这些实例执行正常耗尽，并在其停止后启动替换服务任务。

托管终止	托管耗尽	结果
已禁用	已禁用	Amazon EC2 实例可以随时进行横向缩减或终止，即使其正在运行 Amazon ECS 任务。Amazon ECS 将在替换服务任务停止后启动这些任务。

托管式实例耗尽和竞价型实例耗尽

使用竞价型实例耗尽功能，您可以在 Amazon ECS 代理上设置环境变量 `ECS_ENABLE_SPOT_INSTANCE_DRAINING`，这样 Amazon ECS 就可以将实例置于耗尽状态，以应对两分钟的 Spot 中断。Amazon ECS 托管实例耗尽有助于正常关闭因多种原因而正在终止的 Amazon EC2 实例，而不仅仅是 Spot 中断。例如，您可以使用 Amazon EC2 Auto Scaling 容量再平衡来主动替换中断风险较高的竞价型实例，托管式实例耗尽会正常关闭替换的竞价型实例。使用托管式实例耗尽功能时，您无需单独启用竞价型实例耗尽功能，因此自动扩缩组用户数据中的 `ECS_ENABLE_SPOT_INSTANCE_DRAINING` 是冗余的。有关竞价型实例耗尽的更多信息，请参阅 竞价型实例。

托管式实例耗尽功能如何与 EventBridge 结合使用

Amazon ECS 托管实例耗尽事件会发布到 Amazon EventBridge，并且 Amazon ECS 会在您账户的默认总线中创建 EventBridge 托管规则以支持托管实例耗尽。您可以将这些事件筛选到 Lambda、Amazon SNS 和 Amazon SQS 等其他 AWS 服务，以进行监控和问题排查。

- Amazon EC2 Auto Scaling 在调用生命周期挂钩时向 EventBridge 发送事件。
- Spot 中断通知已发布到 EventBridge。
- Amazon ECS 会生成错误消息，您可以通过 Amazon ECS 控制台和 API 来检索这些消息。

- EventBridge 内置了重试机制，可以缓解临时故障。

配置 Amazon ECS 容量提供程序，以安全关闭实例

使用 Amazon ECS 控制台和 AWS CLI 创建或更新自动扩缩组容量提供程序时，您可以开启托管式实例耗尽功能。

> **Note**
>
> 创建容量提供程序时，默认情况下，托管式实例耗尽功能已开启。

以下是使用 AWS CLI 创建启用托管实例耗尽功能的容量提供程序，以及为集群的现有容量提供程序启用托管实例耗尽功能的示例。

创建启用托管实例耗尽功能的容量提供程序

要创建启用托管实例耗尽功能的容量提供程序，请使用 `create-capacity-provider` 命令。将 `managedDraining` 参数设置为 `ENABLED`。

```
aws ecs create-capacity-provider \
--name capacity-provider \
--auto-scaling-group-provider '{
  "autoScalingGroupArn": "asg-arn",
  "managedScaling": {
    "status": "ENABLED",
    "targetCapacity": 100,
    "minimumScalingStepSize": 1,
    "maximumScalingStepSize": 1
  },
  "managedDraining": "ENABLED",
  "managedTerminationProtection": "ENABLED",
}'
```

响应：

```
{
    "capacityProvider": {
        "capacityProviderArn": "capacity-provider-arn",
        "name": "capacity-provider",
```

```
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1
            },
            "managedTerminationProtection": "ENABLED"
            "managedDraining": "ENABLED"
        }
    }
}
```

为集群的现有容量提供程序启用托管实例耗尽功能

使用 update-capacity-provider 命令为集群的现有容量提供程序启用托管实例耗尽功能。您会看到 managedDraining 目前显示为 DISABLED 而 updateStatus 显示为 UPDATE_IN_PROGRESS。

```
aws ecs update-capacity-provider \
--name cp-draining \
--auto-scaling-group-provider '{
  "managedDraining": "ENABLED"
}
```

响应：

```
{
    "capacityProvider": {
        "capacityProviderArn": "cp-draining-arn",
        "name": "cp-draining",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-draining-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1,
                "instanceWarmupPeriod": 300
            },
```

```
            "managedTerminationProtection": "DISABLED",
            "managedDraining": "DISABLED" // before update
        },
        "updateStatus": "UPDATE_IN_PROGRESS", // in progress and need describe again to
 find out the result
        "tags": [
        ]
    }
}
```

使用 describe-clusters 命令并包含 ATTACHMENTS。托管实例耗尽 attachment 的 status 为 PRECREATED，总体 attachmentsStatus 为 UPDATING。

```
aws ecs describe-clusters --clusters cluster-name --include ATTACHMENTS
```

响应：

```
{
    "clusters": [
        {
            ...

            "capacityProviders": [
                "cp-draining"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                # new precreated managed draining attachment
                {
                    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "type": "managed_draining",
                    "status": "PRECREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "cp-draining"
                        },
                        {
                            "name": "autoScalingLifecycleHookName",
                            "value": "ecs-managed-draining-termination-hook"
                        }
                    ]
```

```
                },

                ...

            ],
            "attachmentsStatus": "UPDATING"
        }
    ],
    "failures": []
}
```

更新完成后，使用 describe-capacity-providers，您将看到 managedDraining 现在为 ENABLED。

```
aws ecs describe-capacity-providers --capacity-providers cp-draining
```

响应：

```
{
    "capacityProviders": [
        {
            "capacityProviderArn": "cp-draining-arn",
            "name": "cp-draining",
            "status": "ACTIVE",
            "autoScalingGroupProvider": {
                "autoScalingGroupArn": "asg-draning-arn",
                "managedScaling": {
                    "status": "ENABLED",
                    "targetCapacity": 100,
                    "minimumScalingStepSize": 1,
                    "maximumScalingStepSize": 1,
                    "instanceWarmupPeriod": 300
                },
                "managedTerminationProtection": "DISABLED",
                "managedDraining": "ENABLED" // successfully update
            },
            "updateStatus": "UPDATE_COMPLETE",
            "tags": []
        }
    ]
}
```

排查 Amazon ECS 托管式实例耗尽问题

您可能需要对托管式实例耗尽问题进行排查。以下是您在使用它时可能遇到的问题和解决方法的示例。

使用自动扩缩时，实例不会在超过最大实例生命周期后终止。

如果您的实例在使用自动扩缩组时达到并超过了实例的最大生命周期后仍未终止，则可能是因为它们受到了防止横向缩减的保护。您可以关闭托管式终止并允许托管耗尽功能来处理实例回收。

使用 AWS Management Console 创建 Amazon ECS 集群自动扩缩资源

了解如何使用 AWS Management Console 创建用于集群自动扩缩的资源。如果资源需要名称，我们将使用前缀 `ConsoleTutorial` 来确保它们都具有唯一的名称，并使它们易于找到。

主题

- 先决条件
- 步骤 1：创建 Amazon ECS 集群
- 第 2 步：注册任务定义
- 第 3 步：运行任务
- 第 4 步：验证
- 第 5 步：清理

先决条件

本教程假设以下先决条件已完成：

- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 创建 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。
- 创建 Amazon ECS 服务相关 IAM 角色。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。
- 创建 Auto Scaling 服务相关 IAM 角色 有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 的服务相关角色。
- 您已创建要使用的 VPC 和安全组。有关更多信息，请参阅 the section called “创建 Virtual Private Cloud”。

步骤 1：创建 Amazon ECS 集群

请按照以下步骤创建 Amazon ECS 集群。

Amazon ECS 将代表您创建一个 Amazon EC2 Auto Scaling 启动模板和 自动扩缩组，作为 AWS CloudFormation 堆栈的一部分。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群，然后选择创建集群。
3. 在集群配置下方的集群名称中，输入 `ConsoleTutorial-cluster`。
4. 在基础设施下，清除 AWS Fargate（无服务器），然后选择 Amazon EC2 实例。接下来，配置充当容量提供程序的自动扩缩组。
   - 在自动扩缩组（ASG）下。选择创建新 ASG，然后提供有关该组的以下详细信息：
     - 对于操作系统/架构，选择 Amazon Linux 2。
     - 对于 EC2 实例类型，选择 t3.nano。
     - 对于 Capacity（容量），输入 自动扩缩组中启动的实例数的最小值和最大值。
5. （可选）要管理集群标签，请展开 Tags（标签），然后执行以下操作之一：

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：

   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。

   [删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。
6. 选择创建。

第 2 步：注册任务定义

您必须先注册任务定义，然后才能在集群上运行任务。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，它使用 Docker Hub 中的 `amazonlinux` 映像，并且直接休眠。有关可用任务定义参数的更多信息，请参阅 Amazon ECS 任务定义。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。

3. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
4. 在 JSON 编辑器框中，粘贴以下内容。

```
{
    "family": "ConsoleTutorial-taskdef",
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "amazonlinux:2",
            "memory": 20,
            "essential": true,
            "command": [
                "sh",
                "-c",
                "sleep infinity"
            ]
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
```

5. 选择创建。

第 3 步：运行任务

为您的账户注册任务定义后，您可以在集群中运行任务。在本教程中，您将在 ConsoleTutorial-cluster 集群中运行五个 ConsoleTutorial-taskdef 任务定义实例。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在集群页面上，选择 ConsoleTutorial-cluster。
3. 在任务下，选择运行新任务。
4. 在环境部分的计算选项下，选择容量提供程序策略。
5. 在部署配置下，对于应用程序类型，选择任务。
6. 从系列下拉列表中选择 ConsoleTutorial-taskdef。
7. 在所需任务下，输入 5。
8. 选择创建。

第 4 步：验证

本教程到目前为止，您应该有一个运行五个任务的集群和一个带有容量提供程序的自动扩缩组。容量提供程序已启用 Amazon ECS 托管扩展。

我们可以通过查看 CloudWatch 指标、自动扩缩组设置和 Amazon ECS 集群任务计数来确认是否一切正常。

查看集群的 CloudWatch 指标

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在屏幕顶部的导航栏中，选择 区域。
3. 在导航窗格中，在指标下，选择所有指标。
4. 在所有指标页面的浏览选项卡下，选择 AWS/ECS/ManagedScaling。
5. 选择 CapacityProviderName, ClusterName。
6. 选中与 ConsoleTutorial-cluster 集群名称对应的复选框。
7. 在图形化指标选项卡下，将周期更改为 30 秒，将统计数据 更改为最大。

   图中的值显示了容量提供程序的目标容量值。它应该从 100 开始，这是我们设定的目标容量百分比。您应该看到它扩展到 200，这将触发目标跟踪调整策略的警报。然后，警报将触发 自动扩缩组扩展。

按照以下步骤查看您的 自动扩缩组详细信息，以确认执行了扩展操作。

确认 自动扩缩组已扩展

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在屏幕顶部的导航栏中，选择 区域。
3. 在导航窗格的 Auto Scaling 下，选择 自动扩缩组。
4. 选择在本教程中创建的 ConsoleTutorial-cluster 自动扩缩组。查看所需容量下的值，然后在实例管理选项卡下查看实例，以确认您的组已横向扩展到两个实例。

按照以下步骤查看您的 Amazon ECS 集群，以确认 Amazon EC2 实例已注册到集群，并且您的任务已转变为 RUNNING 状态。

要验证 自动扩缩组中的实例

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择 `ConsoleTutorial-cluster` 集群。
4. 在任务选项卡上，确认您看到五个处于 `RUNNING` 状态的任务。

第 5 步：清理

完成本教程后，请清除与本教程关联的资源，以避免对您未使用的资源产生费用。不支持删除容量提供程序和任务定义，但这些资源不会产生任何开销。

清除教程资源

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在集群页面上，选择 ConsoleTutorial-cluster。
4. 在 ConsoleTutorial-cluster 页面上，选择任务选项卡，然后选择停止、全部停止。
5. 在导航窗格中，选择集群。
6. 在集群页面上，选择 ConsoleTutorial-cluster。
7. 在页面的右上角，选择删除集群。
8. 在确认框中，输入 delete ConsoleTutorial-cluster，然后选择删除。
9. 按照以下步骤删除 自动扩缩组。
    a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
    b. 在屏幕顶部的导航栏中，选择 区域。
    c. 在导航窗格的 Auto Scaling 下，选择 自动扩缩组。
    d. 选择 `ConsoleTutorial-cluster` 自动扩缩组，然后选择操作。
    e. 从 Actions 菜单中选择 Delete 。在确认框中，输入 delete，然后选择删除。

Amazon ECS 的 Amazon EC2 容器实例

Amazon ECS 容器实例是运行 Amazon ECS 容器代理且注册到集群的 Amazon EC2 实例。当您使用 EC2 启动类型、External 启动类型或自动扩缩组容量提供程序通过 Amazon ECS 运行任务时，您的任务将放置在活动容器实例上。您负责容器实例管理和维护。

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 Amazon EC2 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。

您使用控制台创建集群时，Amazon ECS 会使用与所选操作系统关联的最新 AMI 为您的实例创建启动模板。

使用 AWS CloudFormation 创建集群时，SSM 参数是自动扩缩组实例的 Amazon EC2 启动模板的一部分。您可以将该模板配置为使用动态 Systems Manager 参数来确定要部署的经 Amazon ECS 优化的 AMI。此参数可确保每次部署堆栈时，都会检查是否存在需要应用到 EC2 实例的可用更新。有关如何使用 Systems Manager 参数的示例，请参阅《AWS CloudFormation 用户指南》中的 Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI。

- 检索经 Amazon ECS 优化的 Linux AMI 元数据
- 检索经 Amazon ECS 优化的 Bottlerocket AMI 元数据
- 检索经 Amazon ECS 优化的 Windows AMI 元数据

您可以从与您的应用程序兼容的实例类型中进行选择。使用较大的实例，您可以同时启动更多任务。使用较小的实例，您可以以更精细的方式进行横向扩展以节省成本。您不需要为适合集群中所有应用程序而选择单个 Amazon EC2 实例类型。相反，您可以创建多个自动扩缩组，其中每个组都具有不同的实例类型。然后，您可以为这些组的每一组创建一个 Amazon EC2 容量提供程序。

使用以下指南来确定要使用的实例系列类型和实例类型：

- 消除不满足应用程序特定要求的实例类型或实例系列。例如，如果您的应用程序需要 GPU，则可以排除没有 GPU 的任何实例类型。
- 考虑包括网络吞吐量和存储在内的要求。
- 考虑 CPU 和内存。通常，CPU 和内存必须足够大，从而足以容纳要运行的任务的至少一个副本。

竞价型实例

与按需型实例相比，Spot 容量可以显著节省成本。Spot 容量是定价明显低于按需容量或预留容量的过剩容量。Spot 容量适用于批处理和机器学习工作负载，以及开发和暂存环境。更笼统地说，它适用于可容忍临时停机的任何工作负载。

请理解以下结果，因为 Spot 容量可能并非一直可用。

- 在需求极高的时期，Spot 容量可能不可用。此操作可能会导致 Amazon EC2 竞价型实例的启动被延迟。在这些情况下，Amazon ECS 服务会重试启动任务，Amazon EC2 Auto Scaling 组也会重试启动实例，直到所需的容量变得可用为止。Amazon EC2 不会将 Spot 容量替换为按需容量。
- 当总体容量需求增加时，系统可能会在只有两分钟警告的情况下终止竞价型实例和任务。发出警告后，如有必要，在实例完全终止之前，任务应当开始有序关闭。这有助于最大限度地降低出错的可能性。有关正常关闭的更多信息，请参阅使用 ECS 进行正常关闭。

为了帮助最大限度地减少 Spot 容量短缺情况的出现，请考虑以下建议：

- 使用多个区域和可用区 - Spot 容量因区域和可用区域而异。通过在多个区域和可用区中运行工作负载，您可以提高 Spot 可用性。如果可能，请在运行任务和实例的区域中，为所有可用区指定子网。
- 使用多个 Amazon EC2 实例类型 - 当您将混合实例策略与 Amazon EC2 Auto Scaling 一起使用时，会将多个实例类型启动到您的自动扩缩组中。这样可以确保 Spot 容量请求可以在需要时得到满足。为了最大限度地提高可靠性以及最大限度地降低复杂性，请在混合实例策略中使用具有大致相同数量的 CPU 和内存的实例类型。这些实例可以来自不同的代系，也可以是相同基本实例类型的变体。请注意，它们可能会包含您可能不需要的其他功能。此类列表的一个例子可能包括 m4.large、m5.large、m5a.large、m5d.large、m5n.large、m5dn.large 和 m5ad.large。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的具有多个实例类型和购买选项的自动扩缩组。
- 使用经容量优化的 Spot 分配策略 - 借助 Amazon EC2 Spot，您可以在经容量和成本优化的分配策略之间进行选择。如果您在启动新实例时选择经容量优化的策略，Amazon EC2 Spot 会选择在所选可用区中可用性最高的实例类型。此项有助于减少实例在启动后不久便终止的可能性。

有关如何在容器实例上配置竞价型终止通知的信息，请参阅：

- 配置 Amazon ECS Linux 容器实例以接收竞价型实例通知
- 配置 Amazon ECS Windows 容器实例以接收竞价型实例通知

经 Amazon ECS 优化的 Linux AMI

Amazon ECS 提供已根据这些要求和建议进行了预配置的经 Amazon ECS 优化的 AMI，以运行您的容器工作负载。建议对 Amazon EC2 实例使用经 Amazon ECS 优化的 Amazon Linux 2023 AMI，除非应用程序要求使用 Amazon EC2 基于 GPU 的实例、特定的操作系统或在该 AMI 中尚不可用的 Docker 版本。有关 Amazon Linux 2 和 Amazon Linux 2023 实例的信息，请参阅《Amazon Linux 2023 用户指南》中的比较 Amazon Linux 2 和 Amazon Linux 2023。从最新的经 Amazon ECS 优化的 AMI 启动容器实例可确保您收到最新的安全更新和容器代理版本。有关如何启动实例的信息，请参阅 启动 Amazon ECS Linux 容器实例。

您使用控制台创建集群时，Amazon ECS 会使用与所选操作系统关联的最新 AMI 为您的实例创建启动模板。

使用 AWS CloudFormation 创建集群时，SSM 参数是自动扩缩组实例的 Amazon EC2 启动模板的一部分。您可以将该模板配置为使用动态 Systems Manager 参数来确定要部署的经 Amazon ECS 优化的 AMI。此参数可确保每次部署堆栈时，都会检查是否存在需要应用到 EC2 实例的可用更新。有关如何使用 Systems Manager 参数的示例，请参阅《AWS CloudFormation 用户指南》中的 Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI。

如果您需要自定义经 Amazon ECS 优化的 AMI，请参阅 GitHub 上的 经 Amazon ECS 优化的 AMI 生成配方。

经 Amazon ECS 优化的 AMI 的 Linux 变体使用 Amazon Linux 2 AMI 作为其基础。还提供了 Amazon Linux 2 AMI 版本注释。有关更多信息，请参阅 Amazon Linux 2 版本注释。

建议您使用带有 Linux 内核 5.10 的 AMI，因为 Linux 内核 4.14 已于 2024 年 1 月 10 日终止使用。

以下经 Amazon ECS 优化的 AMI 的变体可用于您的 Amazon EC2 实例。

操作系统	AMI	描述	存储配置
Amazon Linux 2023	经 Amazon ECS 优化的 Amazon Linux 2023 AMI	Amazon Linux 2023 是 AWS 的下一代 Amazon Linux。在大多数情况下，推荐用于为您的 Amazon ECS 工作负载启动 Amazon EC2 实例。有关更多信息，请参阅《Amazon Linux 2023 用户指南》中的什么是 Amazon Linux 2023。	预设情况下，经 Amazon ECS 优化的 Amazon Linux 2023 AMI 附带一个 30GiB 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。 经 Amazon ECS 优化的 Amazon Linux 2023 AMI 的默认文件系统是 `xfs`，Docker 使用 `overlay2` 存储驱动程序。有关
Amazon Linux 2023（arm64）	经 Amazon ECS 优化的 Amazon Linux 2023（arm64）AMI	基于 Amazon Linux 2023，建议在为 Amazon ECS 工作负载启动 Amazon EC2 实例时使用此 AMI，	

操作系统	AMI	描述	存储配置
		这些实例由基于 ARM 的 AWS Graviton/Graviton 2 处理器提供支持。有关更多信息，请参阅《Amazon EC2 用户指南》中的 General Purpose Instances。 经 Amazon ECS 优化的 Amazon Linux 2023（arm64）AMI 不随附 AWS CLI 预装。	更多信息，请参阅 Docker 文档中的使用 OverlayFS 存储驱动程序。

操作系统	AMI	描述	存储配置
Amazon Linux 2023（Neuron）	Amazon Linux 2023（Neuron）	基于 Amazon Linux 2023，此 AMI 适用于 Amazon EC2 Inf1、Trn1 或 Inf2 实例。带有预先配置 AWS Inferentia 和 AWS Trainium 驱动程序以及 Docker 的 AWS Neuron 运行时系统，它使得在 Amazon ECS 上运行机器学习推理工作负载变得更加容易。有关更多信息，请参阅适用于 AWS 神经元机器学习工作负载的 Amazon ECS 任务定义。 经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI 不随附 AWS CLI 预装。	

操作系统	AMI	描述	存储配置
Amazon Linux 2	经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI	基于 Amazon Linux 2，在启动 Amazon EC2 实例时使用此 AMI，您需要为 Amazon ECS 工作负载使用 Linux 内核 5.10 而不是内核 4.14。经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI 未预装 AWS CLI。	预设情况下，基于Amazon Linux 2 的经 Amazon ECS 优化的 AMI（经 Amazon ECS 优化的 Amazon Linux 2 AMI、经 Amazon ECS 优化的 Amazon Linux 2（arm64）AMI 和 Amazon ECS GPU 优化型 AMI）随附单个 30 GiB 根卷发货。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。 经 Amazon ECS 优化的 Amazon Linux 2 AMI 的原定设置文件系统是 `xfs`，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。
	经 Amazon ECS 优化的 Amazon Linux 2 AMI	此项适用于您的 Amazon ECS 工作负载。经 Amazon ECS 优化的 Amazon Linux 2 AMI 不随附 AWS CLI 预装。	
Amazon Linux 2（arm64）	经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10（arm64）AMI	基于 Amazon Linux 2，此 AMI 用于您的 Amazon EC2 实例，这些实例由基于 ARM 的 AWS Graviton/Graviton 2 处理器提供支持，您需要为 Amazon ECS 工作负载使用 Linux 内核 5.10 而不是 Linux 内核 4.14。有关更多信息，请参阅《Amazon EC2 用户指南》中的 General Purpose Instances。	

操作系统	AMI	描述	存储配置
		经 Amazon ECS 优化的 Amazon Linux 2 (arm64) AMI 不随附 AWS CLI 预装。	
	经 Amazon ECS 优化的 Amazon Linux 2（arm64）AMI	基于 Amazon Linux 2，在为 Amazon ECS 工作负载启动 Amazon EC2 实例时使用此 AMI，这些实例由基于 ARM 的 AWS Graviton/Graviton 2 处理器提供支持。 经 Amazon ECS 优化的 Amazon Linux 2 (arm64) AMI 不随附 AWS CLI 预装。	

操作系统	AMI	描述	存储配置
Amazon Linux 2（GPU）	Amazon ECS GPU 优化型内核 5.10 AMI	基于 Amazon Linux 2，建议在为 Amazon ECS 工作负载启动基于 Amazon EC2 GPU 的实例（含 Linux 内核 5.10）时使用此 AMI。预配置了 NVIDIA 内核驱动程序和 Docker GPU 运行时，可以使正在运行的工作负载充分利用 Amazon ECS 上的 GPU。有关更多信息，请参阅 适用于 GPU 工作负载的 Amazon ECS 任务定义。	

操作系统	AMI	描述	存储配置
	Amazon ECS GPU 优化型 AMI	基于 Amazon Linux 2，建议在为 Amazon ECS 工作负载启动基于 Amazon EC2 GPU 的实例（含 Linux 内核 4.14）时使用此 AMI。预配置了 NVIDIA 内核驱动程序和 Docker GPU 运行时，可以使正在运行的工作负载充分利用 Amazon ECS 上的 GPU。有关更多信息，请参阅 适用于 GPU 工作负载的 Amazon ECS 任务定义。	

操作系统	AMI	描述	存储配置
Amazon Linux 2（Neuron）	经 Amazon ECS 优化的 Amazon Linux 2（Neuron）内核 5.10 AMI	基于 Amazon Linux 2，此 AMI 适用于 Amazon EC2 Inf1、Trn1 或 Inf2 实例。其带有预先配置 AWS Inferentia（含 Linux 内核 5.10）和 AWS Trainium 驱动程序以及 Docker 的 AWS Neuron 运行时，使得在 Amazon ECS 上运行机器学习推理工作负载变得更加容易。有关更多信息，请参阅 适用于 AWS 神经元机器学习工作负载的 Amazon ECS 任务定义。经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI 不随附 AWS CLI 预装。	

操作系统	AMI	描述	存储配置
	经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI	基于 Amazon Linux 2，此 AMI 适用于 Amazon EC2 Inf1、Trn1 或 Inf2 实例。带有预先配置 AWS Inferentia 和 AWS Trainium 驱动程序以及 Docker 的 AWS Neuron 运行时系统，它使得在 Amazon ECS 上运行机器学习推理工作负载变得更加容易。有关更多信息，请参阅适用于 AWS 神经元机器学习工作负载的 Amazon ECS 任务定义。经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI 不随附 AWS CLI 预装。	

Amazon ECS 为 GitHub 上的 Amazon ECS 优化 AMI 的 Linux 变体提供了更改日志。有关更多信息，请参阅更改日志。

经 Amazon ECS 优化的 AMI 的 Linux 变体使用 Amazon Linux 2 AMI 或 Amazon Linux 2023 AMI 作为其基础。您可以通过查询 Systems Manager Parameter Store API 来检索每个变体的 Amazon Linux 2 来源 AMI 名称或 Amazon Linux 2023 AMI 名称。有关更多信息，请参阅 检索经 Amazon ECS 优化的 Linux AMI 元数据。还提供了 Amazon Linux 2 AMI 版本注释。有关更多信息，请参阅 Amazon Linux 2 版本注释。还提供了 Amazon Linux 2023 发布说明。有关更多信息，请参阅 Amazon Linux 2023 发布说明。

以下页面提供了有关更改的更多信息：

- GitHub 上的来源 AMI 发布说明

- Docker 文档中的 Docker 引擎版本注释
- NVIDIA 文档中的 NVIDIA 驱动程序文档
- GitHub 上的 Amazon ECS 代理变更日志

  `ecs-init` 应用程序的源代码以及用于打包代理的脚本和配置现在已成为代理存储库的一部分。有关 `ecs-init` 的旧版本和包装，请参阅 GitHub 上的 Amazon ecs-init 变更日志

将安全更新应用于经 Amazon ECS 优化的 AMI

基于 Amazon Linux 的、经 Amazon ECS 优化的 AMI 包含的自定义版本的 cloud-init。Cloud-init 是一个程序包，用于在云计算环境中引导 Linux 映像，并在启动实例时执行所需操作。默认情况下，在 2024 年 6 月 12 日之前发布的所有经 Amazon ECS 优化的 Amazon Linux AMI 在实例启动时都已应用了“关键”和“重要”安全更新。

从 2024 年 6 月 12 日起发布的经 Amazon ECS 优化的 Amazon Linux 2 AMI 开始，默认行为将不再包括在启动时更新程序包。相反，建议您在发布版本后更新为新的经 Amazon ECS 优化的 AMI。经 Amazon ECS 优化的 AMI 在有可用安全更新或基本 AMI 更改时发布。这将确保您收到最新的程序包版本和安全更新，并且程序包版本在实例启动后不可变。有关检索最新经 Amazon ECS 优化 AMI 的更多信息，请参阅 检索经 Amazon ECS 优化的 Linux AMI 元数据。

建议将您的环境自动化，以在新 AMI 可用时进行更新。有关可用选项的信息，请参阅 Amazon ECS enables easier EC2 capacity management, with managed instance draining。

要继续对某个 AMI 版本手动应用“关键”和“重要”安全更新，则可以在 Amazon EC2 实例上运行以下命令。

```
yum update --security
```

如果您想在启动时重新启用安全更新，则可以在启动 Amazon EC2 实例时将以下行添加到 cloud-init 用户数据的 `#cloud-config` 部分。有关更多信息，请参阅《Amazon Linux User Guide》中的 Using cloud-init on Amazon Linux 2。

```
#cloud-config
repo_upgrade: security
```

检索经 Amazon ECS 优化的 Linux AMI 元数据

您可以通过编程方式检索经 Amazon ECS 优化的 AMI 元数据。元数据包括 AMI 名称、Amazon ECS 容器代理版本和 Amazon ECS 运行时版本（其中包括 Docker 版本）。

您使用控制台创建集群时，Amazon ECS 会使用与所选操作系统关联的最新 AMI 为您的实例创建启动模板。

使用 AWS CloudFormation 创建集群时，SSM 参数是自动扩缩组实例的 Amazon EC2 启动模板的一部分。您可以将该模板配置为使用动态 Systems Manager 参数来确定要部署的经 Amazon ECS 优化的 AMI。此参数可确保每次部署堆栈时，都会检查是否存在需要应用到 EC2 实例的可用更新。有关如何使用 Systems Manager 参数的示例，请参阅《AWS CloudFormation 用户指南》中的 Create an Amazon ECS cluster with the Amazon ECS-optimized Amazon Linux 2023 AMI。

经 Amazon ECS 优化的 AMI 的 AMI ID、映像名称、操作系统、容器代理版本、源映像名称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**
>
> 您的管理用户必须具有以下 IAM 权限才能检索经 Amazon ECS 优化的 AMI 元数据。这些权限已添加到 `AmazonECS_FullAccess` IAM policy。
>
> - ssm:GetParameters
> - ssm:GetParameter
> - ssm:GetParametersByPath

Systems Manager Parameter Store 参数格式

以下是经 Amazon ECS 优化的 AMI 变体参数名称的格式。

Linux 经 Amazon ECS 优化的 AMI

- Amazon Linux 2023 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/<version>
```

- Amazon Linux 2023（arm64）AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/<version>
```

- Amazon Linux 2023（Neuron）AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/<version>
```

- Amazon Linux 2 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/<version>
```

- Amazon Linux 2 内核 5.10 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/<version>
```

- Amazon Linux 2 (arm64) AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazal2023neuronamion-linux-2/arm64/<version>
```

- Amazon Linux 2 内核 5.10（arm64）AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/arm64/<version>
```

- Amazon ECS GPU 优化型内核 5.10 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/gpu/<version>
```

- Amazon Linux 2 (GPU) AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/<version>
```

- 经 Amazon ECS 优化的 Amazon Linux 2（Neuron）内核 5.10 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/inf/<version>
```

- Amazon Linux 2（Neuron）AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/inf/<version>
```

以下参数名称格式通过使用子参数 `image_id` 检索 经 Amazon ECS 优化的Amazon Linux 2 AMI 的映像 ID。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

以下参数名称格式通过指定 AMI 名称来检索特定的经 Amazon ECS 优化的 AMI 版本的元数据。

- 经 Amazon ECS 优化的 Amazon Linux 2 AMI 元数据：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

> **Note**
>
> 经 Amazon ECS 优化的 Amazon Linux 2 AMI 的所有版本都可用于检索。只能检索经 Amazon ECS 优化的 AMI 版本 `amzn-ami-2017.09.l-amazon-ecs-optimized` (Linux) 及更高版本。

示例

以下示例说明了可用于检索经 Amazon ECS 优化的 AMI 变体的元数据的方法。

检索最新稳定的经 Amazon ECS 优化的 AMI 的元数据

您可以使用 AWS CLI 和以下 AWS CLI 命令检索最新版稳定的经 Amazon ECS 优化的 AMI。

Linux经 Amazon ECS 优化的 AMI

- 对于经 Amazon ECS 优化的 Amazon Linux 2023 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2023（arm64）AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10（arm64）AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/arm64/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazal2023neuronamion-
linux-2/arm64/recommended --region us-east-1
```

- 对于 Amazon ECS GPU 优化型内核 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/gpu/recommended --region us-east-1
```

- 对于 Amazon ECS GPU 优化型 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2（Neuron）内核 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/inf/recommended --region us-east-1
```

- 对于经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended --region us-east-1
```

检索最新推荐的经 Amazon ECS 优化的 Amazon Linux 2023 AMI 的映像 ID

您可以通过使用子参数 `image_id` 检索最新推荐的经 Amazon ECS 优化的 Amazon Linux 2023 AMI ID 的映像 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-
linux-2023/recommended/image_id --region us-east-1
```

要仅检索 image_id 值，您可以查询特定参数值；例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

检索特定经 Amazon ECS 优化的 Amazon Linux 2 AMI 版本的元数据

使用 AWS CLI 和以下 AWS CLI 命令检索特定 经 Amazon ECS 优化的 Amazon Linux AMI版本的元数据。将要检索的 经 Amazon ECS 优化的 Amazon Linux AMI 的名称替换为 AMI 名称

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-
ecs-hvm-2.0.20200928-x86_64-ebs --region us-east-1
```

使用 Systems Manager GetParametersByPath API 检索经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI 元数据

通过 AWS CLI 和下列命令使用 Systems Manager GetParametersByPath API 检索 经 Amazon ECS 优化的 Amazon Linux 2 AMI 元数据。

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/ --region us-east-1
```

检索最新推荐的经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI 的映像 ID

您可以通过使用子参数 image_id 检索最新推荐的经 Amazon ECS 优化的 Amazon Linux 2 内核 5.10 AMI ID 的映像 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended/image_id --region us-east-1
```

要仅检索 image_id 值，您可以查询特定参数值；例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

在 AWS CloudFormation 模板中使用最新推荐的经 Amazon ECS 优化的 AMI

您可以参考 Systems Manager 参数存储名称引用 AWS CloudFormation 模板中最新推荐的经 Amazon ECS 优化的 AMI。

Linux 示例

```
Parameters:kernel-5.10
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended/
image_id
```

经 Amazon ECS 优化的 Linux AMI 构建脚本

Amazon ECS 已对用于构建 Amazon ECS 优化版 AMI 的 Linux 变体的构建脚本进行开源。GitHub 上现在提供了这些生成脚本。有关更多信息，请参阅 GitHub 上的 amazon-ecs-ami。

如果您需要自定义经 Amazon ECS 优化的 AMI，请参阅 GitHub 上的 经 Amazon ECS 优化的 AMI 生成配方。

生成脚本存储库包含 HashiCorp packer 模板和生成脚本以生成经 Amazon ECS 优化的 AMI 的每个 Linux 变体。这些脚本是经 Amazon ECS 优化的 AMI 生成的可信来源，因此您可关注 GitHub 存储库以监控对 AMI 所做的更改。例如，您可能希望自己的 AMI 使用 Amazon ECS 团队用于正式 AMI 的同一版本的 Docker。

有关更多信息，请通过 GitHub 上的 aws/amazon-ecs-ami 参阅 Amazon ECS AMI 存储桶。

要构建经 Amazon ECS 优化的 Linux AMI

1. 克隆 `aws/amazon-ecs-ami` GitHub 存储库。

   ```
   git clone https://github.com/aws/amazon-ecs-ami.git
   ```

2. 添加 AWS 区域的环境变量以在创建 AMI 时使用。用要使用的区域替换 `us-west-2` 值。

   ```
   export REGION=us-west-2
   ```

3. 提供了一个 Makefile 来构建 AMI。从克隆存储库的根目录中，使用以下命令之一，这对应于要构建的经 Amazon ECS 优化 AMI 的 Linux 变体。

- 经 Amazon ECS 优化的 Amazon Linux 2 AMI

```
make al2
```

- 经 Amazon ECS 优化的 Amazon Linux 2（arm64）AMI

```
make al2arm
```

- Amazon ECS GPU 优化型 AMI

```
make al2gpu
```

- 经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI

```
make al2inf
```

- 经 Amazon ECS 优化的 Amazon Linux 2023 AMI

```
make al2023
```

- 经 Amazon ECS 优化的 Amazon Linux 2023（arm64）AMI

```
make al2023arm
```

- 经 Amazon ECS 优化的 Amazon Linux 2023（Neuron）AMI

```
make al2023neu
```

经 Amazon ECS 优化的 Bottlerocket AMI

Bottlerocket 是一个基于 Linux 的开源操作系统，由 AWS 专门为在虚拟机或裸机主机上运行容器而打造。经 Amazon ECS Bottlerocket 优化的 AMI 安全，只包含运行容器所需的最少软件包数量。此项可以提高资源使用率，减少安全攻击面，并有助于降低管理开销。Bottlerocket AMI 还与 Amazon ECS 集成，以帮助减少更新集群中的容器实例所涉及的操作开销。

Bottlerocket 与 Amazon Linux 存在以下差异：

- Bottlerocket 不包括包管理器，其软件只能作为容器运行。对 Bottlerocket 的更新既可以应用，也可以通过一个步骤进行回滚，这样可以降低出现更新错误的可能性。

- 管理 Bottlerocket 主机的主要机制是使用容器调度器。与 Amazon Linux 不同，登录单个 Bottlerocket 实例的操作并不频繁，仅用于高级调试和故障排除目的。

有关 Bottlerocket 的更多信息，请参阅 GitHub 上的文档和发布。

适用于内核 6.1 和内核 5.10 的经 Amazon ECS 优化的 Bottlerocket AMI 有多个变体。

以下变体使用内核 6.1：

- `aws-ecs-2`
- `aws-ecs-2-nvidia`

以下变体使用内核 5.1.10：

- `aws-ecs-1`
- `aws-ecs-1-nvidia`

  有关 `aws-ecs-1-nvidia` 变体的更多信息，请参阅宣布对 Amazon ECS 上的 Bottlerocket 提供 NVIDIA GPU 支持。

注意事项

将 Bottlerocket AMI 与 Amazon ECS 结合使用时，请考虑以下因素。

- Bottlerocket 支持采用 `x86_64` 和 `arm64` 处理器的 Amazon EC2 实例。Bottlerocket 不建议将 Amazon EC2 实例与 Inferentia 芯片一起使用。
- Bottlerocket 映像不附带 SSH 服务器或 Shell。但是，您可以使用带外管理工具获得 SSH 管理员访问权限并执行引导程序。有关更多信息，请参阅 GitHub 上的 bottlerocket README.md：
  - Exploration (探索)
  - 管理员容器
- 预设情况下，Bottlerocket 启用了一个控制容器。该容器运行 AWS Systems Manager 代理，您可以用它在 Amazon EC2 Bottlerocket 实例上运行命令或启动 shell 会话。有关更多信息，请参阅 AWS Systems Manager 用户指南中的设置会话管理器。
- Bottlerocket 针对容器工作负载进行了优化，并专注于安全性。Bottlerocket 不包含软件包管理器，并且不可变。有关安全功能和指南的信息，请参阅 GitHub 上的安全功能和安全指南。

- Bottlerocket AMI 版本 `1.1.0` 或更高版本支持 `awsvpc` 网络模式。
- Bottlerocket AMI 版本 `1.15.0` 或更高版本支持任务定义中的 App Mesh。
- Bottlerocket AMI 版本 `1.19.0` 或更高版本支持 `initProcessEnabled` 任务定义参数。
- Bottlerocket AMI 也不支持以下服务和功能：
  - ECS Anywhere
  - Service Connect
  - Amazon EFS 处于加密模式和 `awsvpc` 网络模式
  - Elastic Inference 加速器

检索经 Amazon ECS 优化的 Bottlerocket AMI 元数据

您可以通过查询 AWS Systems Manager Parameter Store API 来检索经 Amazon EKS 优化的 AMI 的亚马逊机器映像（AMI）ID。使用此参数，您无需手动查找经 Amazon ECS 优化的 AMI ID。有关 Systems Manager Parameter Store API 的更多信息，请参阅 GetParameter。您使用的用户必须具有 `ssm:GetParameter` IAM 权限才能检索经 Amazon ECS 优化的 AMI 元数据。

`aws-ecs-2` Bottlerocket AMI 变体

您可以使用 AWS CLI 或 AWS Management Console 按 AWS 区域 和架构检索最新的稳定 `aws-ecs-2` Bottlerocket AMI 变体。

- AWS CLI – 您可以使用子参数 `image_id`，通过以下 AWS CLI 命令检索推荐的最新的经 Amazon ECS 优化的 Bottlerocket AMI 的镜像 ID。将 *`region`* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。要检索最新版本以外的版本，请将 `latest` 替换为相应的版本号。
  - 对于 64 位（`x86_64`）架构：

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - 对于 64 位 Arm（`arm64`）架构：

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console – 您可以使用 AWS Management Console 中的 URL 查询推荐的经 Amazon ECS 优化的 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控

制台。在以下 URL 中，将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。

- 对于 64 位（x86_64）架构：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id/description?region=region#
```

- 对于 64 位 Arm（arm64）架构：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id/description?region=region#
```

aws-ecs-2-nvidia Bottlerocket AMI 变体

您可以使用 AWS CLI 或 AWS Management Console 按区域和架构检索最新的稳定 aws-ecs-2-nvdia Bottlerocket AMI 变体。

- AWS CLI – 您可以使用子参数 image_id，通过以下 AWS CLI 命令检索推荐的最新的经 Amazon ECS 优化的 Bottlerocket AMI 的镜像 ID。将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。要检索最新版本以外的版本，请将 latest 替换为相应的版本号。
  - 对于 64 位（x86_64）架构：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 对于 64 位 Arm（arm64）架构：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console – 您可以使用 AWS Management Console 中的 URL 查询推荐的经 Amazon ECS 优化的 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。
  - 对于 64 位（x86_64）架构：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id/description?region=region#
```

- 对于 64 位 Arm（arm64）架构：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id/description?region=region#
```

aws-ecs-1 Bottlerocket AMI 变体

您可以使用 AWS CLI 或 AWS Management Console 按 AWS 区域 和架构检索最新的稳定 aws-ecs-1 Bottlerocket AMI 变体。

- AWS CLI – 您可以使用子参数 image_id，通过以下 AWS CLI 命令检索推荐的最新的经 Amazon ECS 优化的 Bottlerocket AMI 的镜像 ID。将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。要检索最新版本以外的版本，请将 latest 替换为相应的版本号。
  - 对于 64 位（x86_64）架构：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 对于 64 位 Arm（arm64）架构：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-1/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console – 您可以使用 AWS Management Console 中的 URL 查询推荐的经 Amazon ECS 优化的 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。
  - 对于 64 位（x86_64）架构：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/x86_64/latest/image_id/description
```

  - 对于 64 位 Arm（arm64）架构：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-1/arm64/latest/image_id/description
```

aws-ecs-1-nvidia Bottlerocket AMI 变体

您可以使用 AWS CLI 或 AWS Management Console 按区域和架构检索最新的稳定 aws-ecs-1-nvdia Bottlerocket AMI 变体。

- AWS CLI – 您可以使用子参数 image_id，通过以下 AWS CLI 命令检索推荐的最新的经 Amazon ECS 优化的 Bottlerocket AMI 的镜像 ID。将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。要检索最新版本以外的版本，请将 latest 替换为相应的版本号。
  - 对于 64 位（x86_64）架构：

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - 对于 64 位 Arm（arm64）架构：

    ```
    aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
    ecs-1-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console – 您可以使用 AWS Management Console 中的 URL 查询推荐的经 Amazon ECS 优化的 AMI ID。该 URL 使用参数的 ID 的值打开 Amazon EC2 Systems Manager 控制台。在以下 URL 中，将 *region* 替换为您想要的 AMI ID 所对应的区域代码。有关支持的 AWS 区域 的信息，请参阅 GitHub 上的查找 AMI。
  - 对于 64 位（x86_64）架构：

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/x86_64/latest/image_id/description?region=region#
    ```

  - 对于 64 位 Arm（arm64）架构：

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/
    aws-ecs-1-nvidia/arm64/latest/image_id/description?region=region#
    ```

后续步骤

有关如何在 Amazon ECS 上开始使用 Bottlerocket 操作系统的详细教程，请参阅 GitHub 上的在 Amazon ECS 上使用 Bottlerocket AMI 和 AWS 博客网站上的开始使用 Bottlerocket 和 Amazon ECS。

有关如何启动 Bottlerocket 实例的信息，请参阅启动 Amazon ECS 的 Bottlerocket 实例

启动 Amazon ECS 的 Bottlerocket 实例

您可以启动 Bottlerocket 实例，以便运行容器工作负载。

您可以使用 AWS CLI 启动 Bottlerocket 实例。

1. 创建名为 `userdata.toml` 的文件。此文件会用于实例用户数据。将 *`cluster-name`* 替换为您集群的名称。

```
[settings.ecs]
cluster = "cluster-name"
```

2. 使用 the section called “检索经 Amazon ECS 优化的 Bottlerocket AMI 元数据” 中包含的命令之一获取 Bottlerocket AMI ID。您将在以下步骤中使用此 ID。
3. 运行以下命令来启动 Bottlerocket 实例。请记得替换以下参数：
   - 用您的实例将在其中启动的私有或公有子网的 ID 替换*##*。
   - 将 *`bottlerocket_ami`* 替换为上一步中的 AMI ID。
   - 将 *`t3.large`* 替换为您要使用的实例类型。
   - 将 *`region`* 替换为区域代码。

```
aws ec2 run-instances --key-name ecs-bottlerocket-example \
   --subnet-id subnet \
   --image-id bottlerocket_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications
 'ResourceType=instance,Tags=[{Key=bottlerocket,Value=example}]' \
   --user-data file://userdata.toml \
   --iam-instance-profile Name=ecsInstanceRole
```

4. 运行以下命令，以验证容器实例是否注册到集群。在运行此命令时，请记得替换以下参数：

- 将 *cluster* 替换为您的集群名称。
- 将 *region* 替换为区域代码。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

有关如何在 Amazon ECS 上开始使用 Bottlerocket 操作系统的详细演练，请参阅 GitHub 上的在 Amazon ECS 上使用 Bottlerocket AMI 和 AWS 博客网站上的开始使用 Bottlerocket 和 Amazon ECS。

Amazon ECS Linux 容器实例管理

为 Amazon ECS 工作负载使用 EC2 实例时，您需要负责维护这些实例

管理程序

- 启动 Amazon ECS Linux 容器实例
- 引导启动 Amazon ECS Linux 容器实例以传递数据
- 配置 Amazon ECS Linux 容器实例以接收竞价型实例通知
- 在启动 Amazon ECS Linux 容器实例时运行脚本
- 增加 Amazon ECS Linux 容器实例网络接口
- 预留 Amazon ECS Linux 容器实例内存
- 使用 AWS Systems Manager 远程管理 Amazon ECS 容器实例
- 为 Amazon ECS Linux 容器实例使用 HTTP 代理
- 为您的 Amazon ECS Auto Scaling 组配置预初始化的实例
- 更新 Amazon ECS 容器代理

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。

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。

> **Important**
>
> 可靠指标的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更高版本，该版本包含在经 Amazon ECS 优化的 AMI `20220607` 及更高版本中。
> 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。

启动 Amazon ECS Linux 容器实例

您可以使用 Amazon EC2 控制台创建 Amazon ECS 容器实例。

您可以使用各种方法启动实例，包括 Amazon EC2 控制台、AWS CLI 和 SDK。要了解启动实例的其他方法，请参阅《Amazon EC2 用户指南》中的启动实例。

有关启动向导的更多信息，请参阅《Amazon EC2 用户指南》中的使用新启动实例向导启动实例。

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤。

您可以使用新 Amazon EC2 向导启动实例。启动实例向导指定启动实例所需的启动参数。

实例配置的参数

- 过程
- 名称和标签
- 应用程序和操作系统镜像（亚马逊机器映像）
- 实例类型
- 密钥对（登录）
- Network settings (网络设置)
- 配置存储
- 高级详细信息

过程

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤。

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在屏幕顶部的导航栏中，会显示当前 AWS 区域 [例如，美国东部（俄亥俄）]。选择要在其中启动实例的区域。
3. 从 Amazon EC2 控制台控制面板中，选择启动实例。

名称和标签

实例名称是一个标签，其中密钥为 Name（名称），而值为您指定的名称。您可以为实例、卷和弹性图形添加标签。对于竞价型实例，您只能标记竞价型实例请求。

指定实例名称和其它标签为可选项。

- 对于 Name（名称），为实例输入一个描述性名称。如果您没有指定名称，则可以通过其 ID 标识实例，该 ID 将在您启动实例时自动生成。
- 要添加其它标签，请选择 Add additional tags（添加其它标签）。选择 Add tag（添加标签），然后输入密钥和值，然后选择要标记的资源类型。为每个要添加的其它标签选择 Add tag（添加标签）。

应用程序和操作系统镜像（亚马逊机器映像）

亚马逊机器映像（AMI）中包含了创建实例所需的信息。例如，AMI 可能包含充当 Web 服务器所需的软件，例如 Apache 和您的网站。

使用搜索栏查找由 AWS 发布的合适的经 Amazon ECS 优化的 AMI。

1. 在搜索栏中输入以下术语之一。
   - **`ami-ecs`**
   - 经 Amazon ECS 优化的 AMI 的值。

     有关最新的经 Amazon ECS 优化的 AMI 及其值，请参阅经 Linux Amazon ECS 优化的 AMI。
2. 按 Enter 键。
3. 在 Choose an Amazon Machine Image (AMI)（选择亚马逊云机器镜像（AMI））页面上，选择 AWS Marketplace AMIs选项卡。
4. 从左侧 Refine results（优化结果）窗格中，选择 Amazon Web Services 作为 Publisher（发布者）。
5. 在要使用的 AMI 行上选择 Select（选择）。

   或者，选择右上角的 Cancel（取消）以返回启动实例向导，而不选择 AMI。将选择默认 AMI。确保 AMI 满足 Linux 实例中列出的要求。

实例类型

实例类型定义了实例的硬件配置和大小。更大的实例类型拥有更多的 CPU 和内存。有关更多信息，请参阅实例类型。

- 对于 Instance type（实例类型），请为实例选择实例类型。

  您选择的实例类型决定了可用于运行您的任务的资源。

密钥对（登录）

为 Key pair name（密钥对名称）选择一个现有密钥对，或选择 Create new key pair（创建新密钥对）来新建一个密钥对。

> **Important**
>
> 如果您选择 Proceed without key pair (Not recommended)（在没有密钥对的情况下继续（不推荐））选项，则将无法连接到此实例，除非您选择配置为允许用户以其它方式登录的 AMI。

Network settings (网络设置)

根据需要配置网络设置。

- 联网平台：选择 Virtual Private Cloud (VPC)（虚拟私有云（VPC）），则在 Network interfaces（网络接口）部分中指定子网。
- VPC：选择要在其中创建安全组的现有 VPC。
- 子网：您可以在与可用区、本地扩展区、Wavelength 区域或 Outpost 关联的子网中启动实例。

  要在可用区中启动实例，请选择要在其中启动实例的子网。要创建新子网，请选择 Create new subnet 转到 Amazon VPC 控制台。完成此操作后，返回到启动实例向导并选择“Refresh”（刷新）图标，以便将您的子网加载到列表中。

  要在本地区域中启动实例，请选择您在本地区域中创建的子网。

  要在 Outpost 中启动实例，请在 VPC 中选择与 Outpost 关联的子网。
- 自动分配公有 IP：如果实例应可从互联网进行访问，请验证 Auto-assign Public IP（自动分配公有 IP）字段设置为 Enable（启用）。如果不是，请将此字段设置为禁用。

> **Note**
>
> 容器实例需要访问才能与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 端点或具有公共 IP 地址的容器实例实现。
> 有关接口 VPC 端点的更多信息，请参阅 Amazon ECS 接口 VPC 端点 (AWS PrivateLink)

如果您没有配置接口 VPC 端点，并且您的容器实例没有公有 IP 地址，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 为 Amazon ECS Linux 容器实例使用 HTTP 代理。

- Firewall (security groups)（防火墙（安全组））：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。
  - 要选择现有安全组，请选择 Select existing security group（选择现有安全组），然后选择您在 设置以使用 Amazon ECS 中创建的安全组。

配置存储

您选择的 AMI 包含一个或多个存储卷，包括根卷。您可以指定要附加到实例的其它卷。

您可以使用 Simple（简单）视图。

- Storage type（存储类型）：为您的容器实例配置存储。

  如果您使用的是经 Amazon ECS 优化的 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 Amazon ECS 优化型 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。

高级详细信息

对于Advanced details (高级详细信息)，请展开该部分以查看字段并为实例指定任何其他参数。

- 购买选项：选择 Request Spot instances（请求竞价型实例）以请求竞价型实例。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。

.

- IAM 实例配置文件：选择您的容器实例 IAM 角色。其通常被命名为 `ecsInstanceRole`。

> **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。

- （可选）用户数据：使用用户数据（如 Amazon ECS 容器代理配置 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 Amazon ECS 容器实例配置存储在 Amazon S3 中。

```
#!/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 代理配置参数来添加标签。这是由于标签添加到使用 自动扩缩组启动的 Amazon EC2 实例的方式造成的。

```
#!/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 引导启动 Amazon ECS Linux 容器实例以传递数据。

引导启动 Amazon ECS Linux 容器实例以传递数据

在启动 Amazon EC2 实例时，您可以将用户数据传递到 EC2 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 进程守护程序和 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。

要在使用 Amazon EC2 启动向导时传递此用户数据，请参阅 启动 Amazon ECS Linux 容器实例。

您可以将容器实例配置为在容器代理配置或 Docker 进程守护程序配置中传递数据。

Amazon ECS 容器代理

当容器代理启动时，经 Amazon ECS 优化的 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。

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
ECS_WARM_POOLS_CHECK=true
EOF
```

要设置自定义实例属性，请设置 ECS_INSTANCE_ATTRIBUTES 环境变量。

```
#!/bin/bash
cat <<'EOF' >> ecs.config
ECS_INSTANCE_ATTRIBUTES={"envtype":"prod"}
EOF
```

Docker 进程守护程序

您可以使用 Amazon EC2 用户数据指定 Docker 守护进程配置信息。有关配置选项的更多信息，请参阅 Docker 进程守护程序文档。

在以下示例中，自定义选项添加到 Docker 进程守护程序配置文件中，/etc/docker/daemon.json，然后在启动实例时在用户数据中指定。

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"debug": true}
EOF
systemctl restart docker --no-block
```

在以下示例中，自定义选项添加到 Docker 进程守护程序配置文件中，/etc/docker/daemon.json，然后在启动实例时在用户数据中指定。此示例说明如何在 Docker 进程守护程序配置文件中禁用 docker-proxy。

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"userland-proxy": false}
EOF
systemctl restart docker --no-block
```

配置 Amazon ECS Linux 容器实例以接收竞价型实例通知

当 Spot 价格超过您请求的最高价格或容量不再可用时，Amazon EC2会终止、停止或休眠您的 Spot 实例。Amazon EC2 为终止和停止操作提供两分钟的竞价型实例中断通知。它没有提供休眠操作的两分钟通知。如果在实例上开启了 Amazon ECS 竞价型实例耗尽功能，则 Amazon ECS 会收到竞价型实例中断通知，并将实例置于 DRAINING 状态。

> **Important**
>
> 当 Auto Scaling Capacity Rebalancing 移除实例时，Amazon ECS 不会收到来自 Amazon EC2 的通知。有关更多信息，请参阅 Amazon EC2 Auto Scaling 容器重新平衡。

当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。

竞价型实例耗尽功能在默认情况下处于关闭状态。

您可以在启动实例时开启竞价型实例耗尽功能。将以下脚本添加到用户数据字段。将 *MyCluster* 替换为要向其注册容器实例的集群名称。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

要为现有容器实例开启竞价型实例耗尽

1. 通过 SSH 连接到 Spot 实例。

2. 编辑 /etc/ecs/ecs.config 文件并添加以下内容：

```
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
```

3. 重新启动 ecs 服务。

   - 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI：

```
sudo systemctl restart ecs
```

4. (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 “容器自检”。

```
curl http://localhost:51678/v1/metadata
```

在启动 Amazon ECS Linux 容器实例时运行脚本

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. 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《AWS Identity and Access Management 用户指南》中的修改角色。
2. 使用经 Amazon ECS 优化的 Amazon Linux 2 AMI，启动一个或多个容器实例。启动新的容器实例，并在 EC2 用户数据中使用以下示例脚本。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。

   有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

> **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=on-failure
      RestartSec=30
      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==--
```

3. 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/v2 打开控制台。

   b. 在导航栏中，选择您的集群所在的区域。

   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. 在集群页面上，选择任务，然后选择您的任务。

      您启动的每个容器实例都应运行您的任务。

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。

增加 Amazon ECS Linux 容器实例网络接口

> **Note**
>
> 此功能在 Fargate 上不可用。

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原定设置限制，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 网络模式的任务都需一个 ENI，因此，您通常只能在该实例类型上运行两个此类任务。

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开启 awsvpcTrunking 账户设置时，其他 ENI 将在新启动的容器实例上可用。此配置允许您在每个容器实例上放置更多任务。有关 awsvpcTrunking 账户设置的信息，请参阅通过账户设置访问 Amazon ECS 功能。

例如，具有 awsvpcTrunking 的 c5.large 实例增加了十二个的 ENI 限制。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注销容器实例时被删除。有关更多信息，请参阅 EC2 启动类型的 Amazon ECS 任务联网选项。

注意事项

在使用 ENI 中继功能时，注意以下事项。

- 只有经 Amazon ECS 优化的 AMI 的 Linux 版本，或其他具有容器代理 1.28.1 版本或更高版本以及 ecs-init 软件包版本 1.28.1-2 或更高版本的 Amazon Linux 版本才能支持增加的 ENI 限制。只有您使用经 Amazon ECS 优化的 AMI 的 Linux 版本，则将满足这些要求。目前不支持 Windows 容器。
- 仅在启用 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。

- Amazon EC2 实例必须关闭基于资源的 IPv4 DNS 请求。要禁用此选项，请确保在使用 Amazon EC2 控制台中创建新实例时取消选中 Enable resource-based IPV4 (A record) DNS requests（启用基于资源的 IPV4（A 记录）DNS 请求）选项。要使用 AWS CLI 禁用此选项，使用以下命令：

```
aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
```

- 不支持共享子网中的 Amazon EC2 实例。如果使用这些实例，则它们将无法注册到集群。
- 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。
- 您的 Amazon ECS 任务必须在您的容器实例所在的 Amazon VPC 中启动。如果您的任务与容器实例不在同一 VPC 中，则任务将无法启动并显示属性错误。
- 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。您可以通过描述容器实例以查看 statusReason 字段，该字段描述失败的原因。然后可以手动注销或终止容器实例。一旦容器实例成功注销或终止，Amazon ECS 将删除中继 ENI。

> **Note**
>
> Amazon ECS 会发出容器实例状态更改事件，您可以监控这些事件，以查看过渡到 REGISTRATION_FAILED 状态。有关更多信息，请参阅 Amazon ECS 容器实例状态更改事件。

- 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。
- 如果停止然后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。
- 启用 awsvpcTrunking 后，容器实例会收到一个额外的 ENI，其使用 VPC 的默认安全组，并由 Amazon ECS 管理。

先决条件

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 服务链接角色为 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 Management Console 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。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 角色必须启用 awsvpcTrunking 账户设置。我们建议您创建两个容器实例角色（ecsInstanceRole）。然后，您可以为一个角色启用 awsvpcTrunking 账户设置，并将该角色用于需要 ENI 中继的任务。有关容器实例的更多信息，请参阅Amazon ECS 容器实例 IAM 角色。

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 增加的 Amazon ECS 容器网络接口支持的实例。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。只有您使用经 Amazon ECS 优化的 AMI 的 Linux 版本，则将满足这些要求。有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

> **Important**
>
> Amazon EC2 实例必须关闭基于资源的 IPv4 DNS 请求。要禁用此选项，请确保在使用 Amazon EC2 控制台中创建新实例时取消选中 Enable resource-based IPV4 (A record) DNS requests（启用基于资源的 IPV4（A 记录）DNS 请求）选项。要使用 AWS CLI 禁用此选项，使用以下命令：
>
> ```
> aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
> ```

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，这表明其具有中继网络接口。

- list-attributes (AWS CLI)

```
aws ecs list-attributes \
      --target-type container-instance \
      --attribute-name ecs.awsvpc-trunk-id \
      --cluster cluster_name \
```

```
    --region us-east-1
```

- Get-ECSAttributeList (AWS Tools for Windows PowerShell)

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-id -Region us-east-1
```

增加的 Amazon ECS 容器网络接口支持的实例

下面显示了受支持的 Amazon EC2 实例类型，以及在启用 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。对于每种实例类型的弹性网络接口（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）。

Important

虽然同一实例系列中支持其他实例类型，但不支持 a1.metal、c5.metal、c5a.8xlarge、c5ad.8xlarge、c5d.metal、m5.metal、p3dn.24xla 和 r5d.metal 实例类型。
不支持 c5n、d3、d3en、g3、g3s、g4dn、i3、i3en、inf1、m5dn、m5n、m5zn、mac1、r5b、r5n、r5 和 z1d 实例系列。

主题

- 通用型
- 计算优化型
- 内存优化型
- 存储优化
- 加速计算型
- 高性能计算

通用型

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20
a1.2xlarge	3	40
a1.4xlarge	7	60
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.8xlarge	7	60
m5.12xlarge	7	60
m5.16xlarge	14	120
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40
m5a.4xlarge	7	60
m5a.8xlarge	7	60
m5a.12xlarge	7	60

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m5a.16xlarge	14	120
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.8xlarge	7	60
m5ad.12xlarge	7	60
m5ad.16xlarge	14	120
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.8xlarge	7	60
m5d.12xlarge	7	60
m5d.16xlarge	14	120
m5d.24xlarge	14	120
m5d.metal	14	120
m5n.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m5n.xlarge	3	20
m5n.2xlarge	3	40
m5n.4xlarge	7	60
m5n.8xlarge	7	60
m5n.12xlarge	7	60
m5n.16xlarge	14	120
m5zn.large	2	14
m5zn.xlarge	3	31
m5zn.2xlarge	3	64
m5zn.3xlarge	7	98
m5zn.6xlarge	7	120
m6a.large	2	10
m6a.xlarge	3	20
m6a.2xlarge	3	40
m6a.4xlarge	7	60
m6a.8xlarge	7	90
m6a.12xlarge	7	120
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m6a.48xlarge	14	120
m6a.metal	14	120
m6g.medium	1	4
m6g.large	2	10
m6g.xlarge	3	20
m6g.2xlarge	3	40
m6g.4xlarge	7	60
m6g.8xlarge	7	60
m6g.12xlarge	7	60
m6g.16xlarge	14	120
m6g.metal	14	120
m6gd.medium	1	4
m6gd.large	2	10
m6gd.xlarge	3	20
m6gd.2xlarge	3	40
m6gd.4xlarge	7	60
m6gd.8xlarge	7	60
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m6i.large	2	10
m6i.xlarge	3	20
m6i.2xlarge	3	40
m6i.4xlarge	7	60
m6i.8xlarge	7	90
m6i.12xlarge	7	120
m6i.16xlarge	14	120
m6i.24xlarge	14	120
m6i.32xlarge	14	120
m6i.metal	14	120
m6id.large	2	10
m6id.xlarge	3	20
m6id.2xlarge	3	40
m6id.4xlarge	7	60
m6id.8xlarge	7	90
m6id.12xlarge	7	120
m6id.16xlarge	14	120
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m6idn.large	2	10
m6idn.xlarge	3	20
m6idn.2xlarge	3	40
m6idn.4xlarge	7	60
m6idn.8xlarge	7	90
m6idn.12xlarge	7	120
m6idn.16xlarge	14	120
m6idn.24xlarge	14	120
m6idn.32xlarge	15	120
m6idn.metal	15	120
m6in.large	2	10
m6in.xlarge	3	20
m6in.2xlarge	3	40
m6in.4xlarge	7	60
m6in.8xlarge	7	90
m6in.12xlarge	7	120
m6in.16xlarge	14	120
m6in.24xlarge	14	120
m6in.32xlarge	15	120
m6in.metal	15	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m7a.medium	1	4
m7a.large	2	10
m7a.xlarge	3	20
m7a.2xlarge	3	40
m7a.4xlarge	7	60
m7a.8xlarge	7	90
m7a.12xlarge	7	120
m7a.16xlarge	14	120
m7a.24xlarge	14	120
m7a.32xlarge	14	120
m7a.48xlarge	14	120
m7a.metal-48xl	14	120
m7g.medium	1	4
m7g.large	2	10
m7g.xlarge	3	20
m7g.2xlarge	3	40
m7g.4xlarge	7	60
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m7g.metal	14	120
m7gd.medium	1	4
m7gd.large	2	10
m7gd.xlarge	3	20
m7gd.2xlarge	3	40
m7gd.4xlarge	7	60
m7gd.8xlarge	7	60
m7gd.12xlarge	7	60
m7gd.16xlarge	14	120
m7gd.metal	14	120
m7i.large	2	10
m7i.xlarge	3	20
m7i.2xlarge	3	40
m7i.4xlarge	7	60
m7i.8xlarge	7	90
m7i.12xlarge	7	120
m7i.16xlarge	14	120
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
m7i.metal-48xl	14	120
m7i-flex.large	2	4
m7i-flex.xlarge	3	10
m7i-flex.2xlarge	3	20
m7i-flex.4xlarge	7	40
m7i-flex.8xlarge	7	60
mac2.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12

计算优化型

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60
c5.12xlarge	7	60
c5.18xlarge	14	120
c5.24xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c5a.large	2	10
c5a.xlarge	3	20
c5a.2xlarge	3	40
c5a.4xlarge	7	60
c5a.12xlarge	7	60
c5a.16xlarge	14	120
c5a.24xlarge	14	120
c5ad.large	2	10
c5ad.xlarge	3	20
c5ad.2xlarge	3	40
c5ad.4xlarge	7	60
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.12xlarge	7	60

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c5d.18xlarge	14	120
c5d.24xlarge	14	120
c6a.large	2	10
c6a.xlarge	3	20
c6a.2xlarge	3	40
c6a.4xlarge	7	60
c6a.8xlarge	7	90
c6a.12xlarge	7	120
c6a.16xlarge	14	120
c6a.24xlarge	14	120
c6a.32xlarge	14	120
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40
c6g.4xlarge	7	60
c6g.8xlarge	7	60
c6g.12xlarge	7	60

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c6g.16xlarge	14	120
c6g.metal	14	120
c6gd.medium	1	4
c6gd.large	2	10
c6gd.xlarge	3	20
c6gd.2xlarge	3	40
c6gd.4xlarge	7	60
c6gd.8xlarge	7	60
c6gd.12xlarge	7	60
c6gd.16xlarge	14	120
c6gd.metal	14	120
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60
c6gn.12xlarge	7	60
c6gn.16xlarge	14	120
c6i.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c6i.xlarge	3	20
c6i.2xlarge	3	40
c6i.4xlarge	7	60
c6i.8xlarge	7	90
c6i.12xlarge	7	120
c6i.16xlarge	14	120
c6i.24xlarge	14	120
c6i.32xlarge	14	120
c6i.metal	14	120
c6id.large	2	10
c6id.xlarge	3	20
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120
c6id.32xlarge	14	120
c6id.metal	14	120
c6in.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c6in.xlarge	3	20
c6in.2xlarge	3	40
c6in.4xlarge	7	60
c6in.8xlarge	7	90
c6in.12xlarge	7	120
c6in.16xlarge	14	120
c6in.24xlarge	14	120
c6in.32xlarge	15	120
c6in.metal	15	120
c7a.medium	1	4
c7a.large	2	10
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120
c7a.24xlarge	14	120
c7a.32xlarge	14	120
c7a.48xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c7a.metal-48xl	14	120
c7g.medium	1	4
c7g.large	2	10
c7g.xlarge	3	20
c7g.2xlarge	3	40
c7g.4xlarge	7	60
c7g.8xlarge	7	60
c7g.12xlarge	7	60
c7g.16xlarge	14	120
c7g.metal	14	120
c7gd.medium	1	4
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60
c7gd.16xlarge	14	120
c7gd.metal	14	120
c7gn.medium	1	4

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c7gn.large	2	10
c7gn.xlarge	3	20
c7gn.2xlarge	3	40
c7gn.4xlarge	7	60
c7gn.8xlarge	7	60
c7gn.12xlarge	7	60
c7gn.16xlarge	14	120
c7gn.metal	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120
c7i-flex.large	2	4

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
c7i-flex.xlarge	3	10
c7i-flex.2xlarge	3	20
c7i-flex.4xlarge	7	40
c7i-flex.8xlarge	7	60

内存优化型

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r5.large	2	10
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.16xlarge	14	120
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40
r5a.4xlarge	7	60
r5a.8xlarge	7	60
r5a.12xlarge	7	60

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r5a.16xlarge	14	120
r5a.24xlarge	14	120
r5ad.large	2	10
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60
r5ad.8xlarge	7	60
r5ad.12xlarge	7	60
r5ad.16xlarge	14	120
r5ad.24xlarge	14	120
r5b.16xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.8xlarge	7	60
r5d.12xlarge	7	60
r5d.16xlarge	14	120
r5d.24xlarge	14	120
r5dn.16xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r6a.large	2	10
r6a.xlarge	3	20
r6g.2xlarge	3	40
r6a.4xlarge	7	60
r6a.8xlarge	7	90
r6a.12xlarge	7	120
r6a.16xlarge	14	120
r6a.24xlarge	14	120
r6a.32xlarge	14	120
r6a.48xlarge	14	120
r6a.metal	14	120
r6g.medium	1	4
r6g.large	2	10
r6g.xlarge	3	20
r6g.2xlarge	3	40
r6g.4xlarge	7	60
r6g.8xlarge	7	60
r6g.12xlarge	7	60
r6g.16xlarge	14	120
r6g.metal	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r6gd.medium	1	4
r6gd.large	2	10
r6gd.xlarge	3	20
r6gd.2xlarge	3	40
r6gd.4xlarge	7	60
r6gd.8xlarge	7	60
r6gd.12xlarge	7	60
r6gd.16xlarge	14	120
r6gd.metal	14	120
r6i.large	2	10
r6i.xlarge	3	20
r6i.2xlarge	3	40
r6i.4xlarge	7	60
r6i.8xlarge	7	90
r6i.12xlarge	7	120
r6i.16xlarge	14	120
r6i.24xlarge	14	120
r6i.32xlarge	14	120
r6i.metal	14	120
r6idn.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120
r6idn.16xlarge	14	120
r6idn.24xlarge	14	120
r6idn.32xlarge	15	120
r6idn.metal	15	120
r6in.large	2	10
r6in.xlarge	3	20
r6in.2xlarge	3	40
r6in.4xlarge	7	60
r6in.8xlarge	7	90
r6in.12xlarge	7	120
r6in.16xlarge	14	120
r6in.24xlarge	14	120
r6in.32xlarge	15	120
r6in.metal	15	120
r6id.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r6id.xlarge	3	20
r6gd.2xlarge	3	40
r6id.4xlarge	7	60
r6id.8xlarge	7	90
r6id.12xlarge	7	120
r6id.16xlarge	14	120
r6id.24xlarge	14	120
r6id.32xlarge	14	120
r6id.metal	14	120
r7a.medium	1	4
r7a.large	2	10
r7a.xlarge	3	20
r7a.2xlarge	3	40
r7a.4xlarge	7	60
r7a.8xlarge	7	90
r7a.12xlarge	7	120
r7a.16xlarge	14	120
r7a.24xlarge	14	120
r7a.32xlarge	14	120
r7a.48xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r7a.metal-48xl	14	120
r7g.medium	1	4
r7g.large	2	10
r7g.xlarge	3	20
r7g.2xlarge	3	40
r7g.4xlarge	7	60
r7g.8xlarge	7	60
r7g.12xlarge	7	60
r7g.16xlarge	14	120
r7g.metal	14	120
r7gd.medium	1	4
r7gd.large	2	10
r7gd.xlarge	3	20
r7gd.2xlarge	3	40
r7gd.4xlarge	7	60
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60
r7gd.16xlarge	14	120
r7gd.metal	14	120
r7i.large	2	10

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
r7i.xlarge	3	20
r7i.2xlarge	3	40
r7i.4xlarge	7	60
r7i.8xlarge	7	90
r7i.12xlarge	7	120
r7i.16xlarge	14	120
r7i.24xlarge	14	120
r7i.48xlarge	14	120
r7i.metal-24xl	14	120
r7i.metal-48xl	14	120
r7iz.large	2	10
r7iz.xlarge	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
u7i-12tb.224xlarge	14	120
u7in-16tb.224xlarge	15	120
u7in-24tb.224xlarge	15	120
u7in-32tb.224xlarge	15	120
x2gd.medium	1	10
x2gd.large	2	10
x2gd.xlarge	3	20
x2gd.2xlarge	3	40
x2gd.4xlarge	7	60
x2gd.8xlarge	7	60
x2gd.12xlarge	7	60
x2gd.16xlarge	14	120
x2gd.metal	14	120
x2idn.16xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
x2idn.24xlarge	14	120
x2idn.32xlarge	14	120
x2idn.metal	14	120
x2iedn.xlarge	3	13
x2iedn.2xlarge	3	29
x2iedn.4xlarge	7	60
x2iedn.8xlarge	7	120
x2iedn.16xlarge	14	120
x2iedn.24xlarge	14	120
x2iedn.32xlarge	14	120
x2iedn.metal	14	120
x2iezn.2xlarge	3	64
x2iezn.4xlarge	7	120
x2iezn.6xlarge	7	120
x2iezn.8xlarge	7	120
x2iezn.12xlarge	14	120
x2iezn.metal	14	120

存储优化

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60
i4g.8xlarge	7	60
i4g.16xlarge	14	120
i4i.xlarge	3	8
i4i.2xlarge	3	28
i4i.4xlarge	7	58
i4i.8xlarge	7	118
i4i.12xlarge	7	118
i4i.16xlarge	14	248
i4i.24xlarge	14	118
i4i.32xlarge	14	498
i4i.metal	14	498
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40
im4gn.4xlarge	7	60

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
im4gn.8xlarge	7	60
im4gn.16xlarge	14	120
is4gen.medium	1	4
is4gen.large	2	10
is4gen.xlarge	3	20
is4gen.2xlarge	3	40
is4gen.4xlarge	7	60
is4gen.8xlarge	7	60

加速计算型

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
g5.4xlarge	7	40
g5.8xlarge	7	90
g5.12xlarge	14	120
g5.16xlarge	7	120
g5.24xlarge	14	120
g5.48xlarge	6	120
g5g.xlarge	3	20
g5g.2xlarge	3	40
g5g.4xlarge	7	60
g5g.8xlarge	7	60
g5g.16xlarge	14	120
g5g.metal	14	120
g6.xlarge	3	20
g6.2xlarge	3	40
g6.4xlarge	7	60
g6.8xlarge	7	90
g6.12xlarge	7	120
g6.16xlarge	14	120
g6.24xlarge	14	120
g6.48xlarge	14	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
gr6.4xlarge	7	60
gr6.8xlarge	7	90
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.48xlarge	63	242
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

高性能计算

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
hpc6a.48xlarge	1	120
hpc6id.32xlarge	1	120

实例类型	没有 ENI 中继的任务限制	有 ENI 中继的任务限制
hpc7g.4xlarge	3	120
hpc7g.8xlarge	3	120
hpc7g.16xlarge	3	120

预留 Amazon ECS Linux 容器实例内存

当 Amazon ECS 容器代理将容器实例注册到集群中时，代理必须确定容器实例可为任务预留的内存量。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，这并不总是转换为可用于任务的 8192MiB 内存。

Amazon ECS 容器代理提供配置变量调用的 `ECS_RESERVED_MEMORY`，您可以使用该变量从分配给您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。

如果您的任务占用容器实例上的所有内存，则您的任务可能会与关键系统进程争夺内存，并可能引起系统故障。

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 容器代理配置和引导启动 Amazon ECS Linux 容器实例以传递数据。

如果为任务指定 8192 MiB，并且没有任何容器实例具有 8192 MiB 或以上的可用内存来满足此要求，则任务无法放置在您的集群中。如果使用托管计算环境，则 AWS Batch 必须启动更大的实例类型来满足该要求。

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引起系统故障。

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
```

运行经 Amazon ECS 优化的 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
```

运行经 Amazon ECS 优化的 Windows Server 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。

查看容器实例内存

您可以在 Amazon ECS 控制台中（或使用 DescribeContainerInstances API 操作）查看容器实例注册的内存量。如果您试图通过为特定实例类型的任务提供尽可能多的内存来最大限度地提高资源利用率，则可以观察容器实例的可用内存，然后为您的任务分配足够的内存。

查看容器实例内存

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群，然后选择托管您的容器实例的集群。
3. 选择基础设施，然后在容器实例下选择容器实例。
4. 资源部分显示了容器实例的已注册内存和可用内存。

   注册的内存值是容器实例首次启动时向 Amazon ECS 注册的值，可用内存值是尚未分配给任务的值。

使用 AWS Systems Manager 远程管理 Amazon ECS 容器实例

您可以使用 AWS Systems Manager (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。

有关 Run Command 的更多信息，请参阅 AWS Systems Manager 用户指南中的 AWS Systems ManagerRun Command。

以下是将 Systems Manager 与 Amazon ECS 结合使用的先决条件。

1. 您必须向容器实例角色（ecsInstanceRole）授予访问 Systems Manager API 的权限。为此，您可以将 AmazonSSMManagedInstanceCore 分配给 `ecsInstanceRole` 角色。有关如何将策略附加到角色的信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色权限策略（控制台）
2. 验证您的容器实例上是否安装了 SSM Agent。有关更多信息，请参阅在 Linux EC2 实例上手动安装 SSM Agent。

将 Systems Manager 托管策略附加到您的 `ecsInstanceRole` 并验证容器实例上是否安装了 AWS Systems Manager 代理 (SSM Agent) 后，可以开始使用 Run Command 向容器实例发送命令。有关在实例上运行命令和 shell 脚本以及查看结果输出的信息，请参见 AWS Systems Manager用户手册中的使用 Systems Manager Run Command 运行命令和 Run Command 演练 。

常见的应用场景是使用 Run Command 更新容器实例软件。您可以使用以下参数按照《AWS Systems Manager 用户指南》中的步骤进行操作。

参数	值
命令文档	AWS-RunShellScript
命令	`$ yum update -y`
目标实例	您的容器实例

为 Amazon ECS Linux 容器实例使用 HTTP 代理

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互联网网关、NAT 网关或实例访问外部网络，则这非常有用。

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，然后重新启动代理。

`/etc/ecs/ecs.config`（Amazon Linux 2 和 AmazonLinux 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 Amazon ECS 代理用来连接到互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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` 代理用来连接到互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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 (仅限 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 代理用来连接到互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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=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互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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。

有关如何配置代理的信息，请参阅如何在 Amazon Linux 2 或 AL2023 中为 Docker 和 Amazon ECS 容器代理设置 HTTP 代理？。

为您的 Amazon ECS Auto Scaling 组配置预初始化的实例

Amazon ECS 支持 Amazon EC2 Auto Scaling 暖池。暖池是一组准备投入使用的预初始化 Amazon EC2 实例。每当您的应用程序需要横向扩展时，Amazon EC2 Auto Scaling 都会使用暖池中的预初始化实例，而不是启动冷实例，允许运行任何最终初始化过程，然后将实例投入使用。

要了解有关暖池以及如何将暖池添加到 自动扩缩组的更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 的暖池。

当您为 Amazon ECS 的自动扩缩组创建或更新温池时，无法设置在横向缩减时将实例退回暖池的选项（`ReuseOnScaleIn`）。有关更多信息，请参阅《AWS Command Line Interface 参考》中的 put-warm-pool。

要将暖池与您的 Amazon ECS 集群一起使用，请在 Amazon EC2 Auto Scaling 组启动模板的 User data（用户数据）字段中将 `ECS_WARM_POOLS_CHECK` 代理配置变量设置为 `true`。

以下示例介绍如何在 Amazon EC2 启动模板的 User data（用户数据）字段中指定代理配置变量。将 *`MyCluster`* 替换为您的集群的名称。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_WARM_POOLS_CHECK=true
EOF
```

`ECS_WARM_POOLS_CHECK` 变量仅在代理版本 `1.59.0` 和更高版本上受支持。有关变量的更多信息，请参阅 Amazon ECS 容器代理配置。

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否通过经 Amazon ECS 优化过的 AMI 或其他操作系统启动。

> **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。

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。

Amazon ECS console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择您的外部实例将注册的区域。
3. 在导航窗格中，选择 集群 并选择托管外部实例的集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。
5. 在 Container instances（容器实例）下，注意容器实例的 Agent version（代理版本）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。

   如果您的代理版本已过期，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经 Amazon ECS 优化的 AMI，请参阅 更新经 Amazon ECS 优化的 AMI 上的 Amazon ECS 容器代理。
   - 如果您的容器实例未运行经 Amazon ECS 优化的 AMI，请参阅 手动更新 Amazon ECS 容器代理（适用于非经 Amazon ECS 优化的 AMI）。

   > **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 Linux 容器实例。

Amazon ECS container agent introspection API

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 容器自检。

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 | python3 -mjson.tool
```

> **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。

更新经 Amazon ECS 优化的 AMI 上的 Amazon ECS 容器代理

如果您使用经 Amazon ECS 优化的 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经 Amazon ECS 优化的 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 经 Amazon ECS 优化的 Linux AMI。
- 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 更新经 Amazon ECS 优化过的 AMI 上的 `ecs-init` 程序包。
- 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。

> **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。

更新经 Amazon ECS 优化过的 AMI 上的 **`ecs-init`** 程序包

1. 通过 SSH 登录到容器实例。

2. 使用以下命令更新 ecs-init 包。

```
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：
>
> - 经 Amazon ECS 优化的 Amazon ECS Amazon Linux 2 AMI：
>
>   ```
>   sudo systemctl restart docker
>   ```
>
> - 经 Amazon ECS 优化的 Amazon ECS 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**
>
> UpdateContainerAgent API 仅在经 Amazon ECS 优化的 AMI 的Linux变体上受支持，但 Amazon ECS 优化的 Amazon Linux 2（arm64）AMI 除外。对于使用经 Amazon ECS 优化的 Amazon Linux 2 (arm64) AMI 的容器实例，请更新 ecs-init 程序包更新代理。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（适用于非经 Amazon ECS 优化的 AMI）。如果您使用的是 Windows 容器实例，建议您启动新的容器实例以更新 Windows 群集中的代理版本。

UpdateContainerAgent API 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。Amazon ECS 将对照最新的代理版本检查您当前的代理版本，并检查是否可以更新。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。

> **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。

在控制台中更新经 Amazon ECS 优化的 AMI 上的 Amazon ECS 容器代理

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择您的外部实例将注册的区域。
3. 在导航窗格中，选择 Clusters 并选择集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。
5. 在容器实例下，选择要更新的实例，然后选择操作、更新代理。

手动更新 Amazon ECS 容器代理（适用于非经 Amazon ECS 优化的 AMI）

手动更新 Amazon ECS 容器代理（适用于非经 Amazon ECS 优化的 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。

> **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 Amazon ECS 容器实例配置存储在 Amazon S3 中 和 在 Amazon ECS 中使用非 AWS 容器映像。

7. 从 Amazon Elastic Container Registry Public 拉取最新的 Amazon ECS 容器代理映像。

```
ubuntu:~$ docker pull public.ecr.aws/ecs/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。 经 Amazon ECS 优化的 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。

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。

> **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 \
--volume=/etc/ecs:/etc/ecs/pki \
--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。

经 Amazon ECS 优化的 Windows AMI

经 Amazon ECS 优化的 AMI 预配置了运行 Amazon ECS 工作负载所需的必要组件。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。

经 Amazon ECS 优化的 AMI 元数据，包括 AMI 名称、Amazon ECS 容器代理版本和 Amazon ECS 运行时版本（其中包括 Docker 版本，对于每个变体可以编程方式检索）。有关更多信息，请参阅 the section called “检索经 Amazon ECS 优化的 Windows AMI 元数据”。

您可以订阅 Windows AMI Amazon SNS 主题，以便发布新 AMI 或将 AMI 版本标记为专用时收到通知。有关更多信息，请参阅 订阅经 Amazon ECS 优化的 Windows AMI 更新通知。

> **Important**
>
> 8 月之后生产的所有经 ECS 优化的 AMI 变体都将从 Docker EE（Mirantis）迁移到 Docker CE（Moby 项目）。
> 为了确保客户预设情况下安装了最新的安全更新，Amazon ECS 保留至少三个 Windows 经 Amazon ECS 优化的 AMI。在发布新的 Windows 经 Amazon ECS 优化的 AMI 后，Amazon ECS 将使 Windows 经 Amazon ECS 优化的 AMI 成为旧版专用。如果存在需要访问的私有 AMI，请通过向 Cloud Support 提交服务单来告知我们。

经 Amazon ECS 优化的 AMI 变体

以下 Windows 服务器版本的经 Amazon ECS优化 AMI 适用于您的 Amazon EC2 实例。

> **Important**
>
> 8 月之后生产的所有经 ECS 优化的 AMI 变体都将从 Docker EE（Mirantis）迁移到 Docker CE（Moby 项目）。

- 经 Amazon ECS 优化的 Windows Server 2022 Full AMI
- 经 Amazon ECS 优化的 Windows Server 2022 Core AMI
- 经 Amazon ECS 优化的 Windows Server 2019 Full AMI
- 经 Amazon ECS 优化的 Windows Server 2019 Core AMI
- 经 Amazon ECS 优化的 Windows Server 2016 Full AMI

> **Important**
>
> Windows Server 2016 不支持最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 Full AMI 将不会收到 Docker 运行时的安全补丁或错误补丁。建议您迁移到以下 Windows 平台之一：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

2022 年 8 月 9 日，经 Amazon ECS 优化的 Windows Server 20H2 Core AMI 达到终止支持日期。不会发布此 AMI 的任何新版本。有关更多信息，请参阅 Windows Server 发行版信息。

Windows Server 2022、Windows Server 2019 和 Windows Server 2016 是长期服务渠道（LTSC）版本。Windows Server 20H2 是一个半年服务渠道（SAC）版本。有关更多信息，请参阅 Windows Server 发行版信息。

注意事项

下面是您应了解的有关 Amazon EC2 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，情况也相反。为了更好地放置 Windows 和 Linux 任务，请将 Windows 和 Linux 容器实例保持在单独的集群中，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')`。
- 使用 EC2 和 Fargate 启动类型的任务支持 Windows 容器。
- 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 运行 Windows 的 EC2 实例的 Amazon ECS 任务定义差异。
- 对于任务的 IAM 角色功能，您需要将 Windows 容器实例配置为在启动时允许该功能。容器在使用此功能时必须运行一些提供的 PowerShell 代码。有关更多信息，请参阅 Amazon EC2 Windows 实例附加配置。
- 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 Application Load Balancer 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 使用负载均衡分配 Amazon ECS 服务流量。
- Windows Server Docker 映像很大（9GiB）。因此，您的 Windows 容器实例需要比 Linux 容器实例更多的存储空间。
- 要在 Windows Server 上运行 Windows 容器，容器的基本映像操作系统版本必须与主机的操作系统版本匹配。有关更多信息，请参阅 Microsoft 文档网站上的 Windows 容器版本兼容性。如果您

的集群运行多个 Windows 版本，则可以使用置放约束来确保将任务放在运行相同版本的 EC2 实例上：`memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)`。有关更多信息，请参阅 the section called “检索经 Amazon ECS 优化的 Windows AMI 元数据”。

检索经 Amazon ECS 优化的 Windows AMI 元数据

经 Amazon ECS 优化的 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**
>
> 您的管理用户必须具有以下 IAM 权限才能检索经 Amazon ECS 优化的 AMI 元数据。这些权限已添加到 `AmazonECS_FullAccess` IAM policy。
>
> - ssm:GetParameters
> - ssm:GetParameter
> - ssm:GetParametersByPath

Systems Manager Parameter Store 参数格式

> **Note**
>
> 以下 Systems Manager Parameter Store API 参数已被弃用，不应使用这些参数来检索最新的 Windows AMI：
>
> - `/aws/service/ecs/optimized-ami/windows_server/2016/english/full/recommended/image_id`
> - `/aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id`

以下是经 Amazon ECS 优化的 AMI 变体参数名称的格式。

- Windows Server 2022 Full AMI 元数据：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

- Windows Server 2022 Core AMI 元数据：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

- Windows Server 2019 Full AMI 元数据：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

- Windows Server 2019 Core AMI 元数据：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

- Windows Server 2016 Full AMI 元数据：

```
/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

以下参数名称格式检索最新稳定的 Windows Server 2019 Full AMI 都元数据

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-
English-Full-ECS_Optimized
```

以下是针对参数值返回的 JSON 对象的示例。

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-
ECS_Optimized",
            "Type": "String",
            "Value": "{\"image_name\":\"Windows_Server-2019-English-Full-
ECS_Optimized-2023.06.13\",\"image_id\":\"ami-0debc1fb48e4aee16\",\"ecs_runtime_version
\":\"Docker (CE) version 20.10.21\",\"ecs_agent_version\":\"1.72.0\"}",
            "Version": 58,
            "LastModifiedDate": "2023-06-22T19:37:37.841000-04:00",
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ami-windows-latest/
Windows_Server-2019-English-Full-ECS_Optimized",
            "DataType": "text"
```

```
        }
    ],
    "InvalidParameters": []
}
```

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`

示例

以下示例说明了可用于检索经 Amazon ECS 优化的 AMI 变体的元数据的方法。

检索最新稳定的经 Amazon ECS 优化的 AMI 的元数据

您可以使用 AWS CLI 和以下 AWS CLI 命令检索最新版稳定的经 Amazon ECS 优化的 AMI。

- 对于经 Amazon ECS 优化的 Windows Server 2022 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized --region us-east-1
```

- 对于经 Amazon ECS 优化的 Windows Server 2022 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized --region us-east-1
```

- 对于经 Amazon ECS 优化的 Windows Server 2019 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized --region us-east-1
```

- 对于经 Amazon ECS 优化的 Windows Server 2019 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized --region us-east-1
```

- 对于经 Amazon ECS 优化的 Windows Server 2016 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized --region us-east-1
```

在 AWS CloudFormation 模板中使用最新推荐的经 Amazon ECS 优化的 AMI

您可以参考 Systems Manager 参数存储名称引用 AWS CloudFormation 模板中最新推荐的经 Amazon ECS 优化的 AMI。

```
Parameters:
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized/image_id
```

订阅经 Amazon ECS 优化的 Windows AMI 更新通知

AWS 为与 Windows Server AMI 相关的通知提供了两个 Amazon SNS 主题 ARN。当发布新的 Windows Server AMI 时，一个主题会发送更新通知。另一个主题在以前发布的 Windows Server AMI 变为专用时发送通知。虽然这些主题并不特定于经 Amazon ECS 优化的 Windows AMI，但由于经 Amazon ECS 优化的 Windows AMI 遵循相同的发布计划，因此您可以使用这些通知指示更新新的经 Amazon ECS 优化的 Windows AMI 的时间。有关订阅 Windows AMI 通知的更多信息，请参阅《Amazon EC2 用户指南》中的 Subscribing to Windows AMI notifications。

> **Note**
>
> 您的用户或附加到您的用户的角色必须具有订阅 Amazon SNS 主题的 `sns::subscribe` IAM 权限。

经 Amazon ECS 优化的 Windows AMI 版本

查看当前和以前版本的经 Amazon ECS 优化的 AMI 及其相应版本的 Amazon ECS 容器代理、Docker 和 `ecs-init` 程序包。

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 the section called “检索经 Amazon ECS 优化的 Windows AMI 元数据”。

以下选项卡显示了 Windows 经 Amazon ECS 优化的 AMI 版本列表。有关在 AWS CloudFormation 模板中引用 Systems Manager Parameter Store参数的详细信息，请参阅 在 AWS CloudFormation 模板中使用最新推荐的经 Amazon ECS 优化的 AMI 。

Important

为了确保客户预设情况下安装了最新的安全更新，Amazon ECS 保留至少三个 Windows 经 Amazon ECS 优化的 AMI。在发布新的 Windows 经 Amazon ECS 优化的 AMI 后，Amazon ECS 将使 Windows 经 Amazon ECS 优化的 AMI 成为旧版专用。如果存在需要访问的私有 AMI，请通过向 Cloud Support 提交服务单来告知我们。
Windows Server 2016 不支持最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 Full AMI 将不会收到 Docker 运行时的安全补丁或错误补丁。建议您迁移到以下 Windows 平台之一：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

Windows Server 2022 Full AMI versions

下表列出了当前和以前版本的经 Amazon ECS 优化的 Windows Server 2022 Full AMI 及其相应版本的 Amazon ECS 容器代理和 Docker。

经 Amazon ECS 优化的 Windows Server 2022 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	公开

经 Amazon ECS 优化的 Windows Server 2022 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	公开
Windows_Server-2022-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	公开
Windows_Server-2022-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	公开
Windows_Server-2022-English-Full-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	公开
Windows_Server-2022-English-Full-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2022 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2022 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2022 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	专属
Windows_Server-2022-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	专属
Windows_Server-2022-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	专属
Windows_Server-2022-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	专属
Windows_Server-2022-English-Full-ECS_Optimized-2021.009.23	1.55.3	20.10.7	专属

使用以下 AWS CLI 命令检索当前经 Amazon ECS 优化的 Windows Server 2022 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

Windows Server 2022 Core AMI versions

下表列出了当前和以前版本的经 Amazon ECS 优化的 Windows Server 2022 Core AMI 及其相应版本的 Amazon ECS 容器代理和 Docker。

经 Amazon ECS 优化的 Windows Server 2022 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	公开
Windows_Server-2022-English-Core-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	公开
Windows_Server-2022-English-Core-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	公开
Windows_Server-2022-English-Core-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	公开
Windows_Server-2022-English-Core-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	公开

经 Amazon ECS 优化的 Windows Server 2022 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2022 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2022 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	专属
Windows_Server-2022-English-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	专属
Windows_Server-2022-English-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	专属

经 Amazon ECS 优化的 Windows Server 2022 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	专属
Windows_Server-2022-English-Core-ECS_Optimized-2021.09.23	1.55.3	20.10.7	专属

使用以下 AWS CLI 命令检索当前经 Amazon ECS 优化的 Windows Server 2022 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

Windows Server 2019 Full AMI versions

下表列出了当前和以前版本的经 Amazon ECS 优化的Windows Server 2019 Full AMI 及其相应版本的 Amazon ECS 容器代理和 Docker。

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	公开
Windows_Server-2019-English-Full-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	公开

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	公开
Windows_Server-2019-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	公开
Windows_Server-2019-English-Full-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	公开
Windows_Server-2019-English-Full-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	专属
Windows_Server-2019-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.09.23	1.55.3	20.10.7	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.08.12	1.55.0	20.10.6	公开

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2021.07.13	1.54.02	20.10.6	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.07.08	1.54.0	20.10.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.06.11	1.53.0	20.10.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.05.21	1.52.2	20.10.4	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.04.14	1.51.0	20.10.0	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.03.11	1.50.2	19.03.14	专属
Windows_Server-2019-English-Full-ECS_Optimized-2021.02.10	1.50.0	19.03.14	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2021.01.13	1.49.0	19.03.14	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.11.18	1.48.0	19.03.13	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.11.06	1.47.0	19.03.11	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.10.14	1.45.0	19.03.11	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.08.12	1.43.0	19.03.11	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.07.15	1.41.1	19.03.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.06.11	1.40.0	19.03.5	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2020.05.14	1.39.0	19.03.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2020.01.15	1.35.0	19.03.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.12.16	1.34.0	19.03.5	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.11.25	1.34.0	19.03.4	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.11.13	1.32.1	19.03.4	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.10.09	1.32.0	19.03.2	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.09.11	1.30.0	19.03.1	专属

经 Amazon ECS 优化的 Windows Server 2019 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2019.08.16	1.29.1	19.03.1	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.07.19	1.29.0	18.09.8	专属
Windows_Server-2019-English-Full-ECS_Optimized-2019.05.10	1.27.0	18.09.4	专属

使用以下 AWS CLI 命令检索当前经 Amazon ECS 优化的 Windows Server 2019 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Windows Server 2019 Core AMI versions

> **Important**
>
> 下表列出了当前和以前版本的经 Amazon ECS 优化的Windows Server 2019 Core AMI 及其相应版本的 Amazon ECS 容器代理和 Docker。

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	公开
Windows_Server-2019-English-Core-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	公开
Windows_Server-2019-English-Core-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	公开
Windows_Server-2019-English-Core-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	公开
Windows_Server-2019-English-Core-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	公开

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.08.15	1.62.1	20.10.9	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.07.13	1.61.3	20.10.9	专属
Windows_Server-2019-English-Core-ECS_Optimized-2022.06.15	1.61.2	20.10.9	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2022.01.18	1.57.1	20.10.9	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.12.16	1.57.1	20.10.7	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.11.11	1.57.0	20.10.7	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.09.23	1.55.3	20.10.7	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.08.12	1.55.0	20.10.6	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2021.07.13	1.54.02	20.10.6	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.07.08	1.54.0	20.10.6	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.06.11	1.53.0	20.10.5	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.05.21	1.52.2	20.10.4	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.04.14	1.51.0	20.10.0	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2021.03.11	1.50.2	19.03.14	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.02.10	1.50.0	19.03.14	专属
Windows_Server-2019-English-Core-ECS_Optimized-2021.01.13	1.49.0	19.03.14	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.11.18	1.48.0	19.03.13	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.11.06	1.47.0	19.03.11	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2020.10.14	1.45.0	19.03.11	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.09.09	1.44.3	19.03.11	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.08.12	1.43.0	19.03.11	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.07.15	1.41.1	19.03.5	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.06.11	1.40.0	19.03.5	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2020.05.14	1.39.0	19.03.5	专属
Windows_Server-2019-English-Core-ECS_Optimized-2020.01.15	1.35.0	19.03.5	专属
Windows_Server-2019-English-Core-ECS_Optimized-2019.12.16	1.34.0	19.03.5	专属
Windows_Server-2019-English-Core-ECS_Optimized-2019.11.25	1.34.0	19.03.4	专属
Windows_Server-2019-English-Core-ECS_Optimized-2019.11.13	1.32.1	19.03.4	专属

经 Amazon ECS 优化的 Windows Server 2019 Core AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2019.10.09	1.32.0	19.03.2	专属

使用以下 AWS CLI 命令检索当前经 Amazon ECS 优化的 Windows Server 2019 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

Windows Server 2016 Full AMI versions

Important

Windows Server 2016 不支持最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 Full AMI 将不会收到 Docker 运行时的安全补丁或错误补丁。建议您迁移到以下 Windows 平台之一：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

下表列出了当前和以前版本的经 Amazon ECS 优化的Windows Server 2016 Full AMI 及其相应版本的 Amazon ECS 容器代理和 Docker。

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	公开
Windows_Server-2016-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	公开
Windows_Server-2016-English-Full-ECS_Optimized-2024.01.09	1.79.2	20.10.23 (Docker CE)	公开
Windows_Server-2016-English-Full-ECS_Optimized-2023.12.12	1.79.1	20.10.23 (Docker CE)	公开
Windows_Server-2016-English-Full-ECS_Optimized-2023.11.14	1.79.0	20.10.23 (Docker CE)	公开
Windows_Server-2016-English-Full-ECS_Optimized-2023.10.11	1.77.0	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.09.15	1.75.3	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2023.08.09	1.74.1	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.07.11	1.73.1	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.06.13	1.72.0	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.05.18	1.71.1	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.04.18	1.70.2	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.03.21	1.69.0	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2023.02.21	1.68.2	20.10.21 (Docker CE)	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2023.01.11	1.68.0	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.12.14	1.67.2	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.11.09	1.65.1	20.10.21 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.10.12	1.64.0	20.10.17 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.09.22	1.63.1	20.10.17 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.09.14	1.62.2	20.10.17 (Docker CE)	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.08.15	1.62.1	20.10.9	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2022.07.13	1.61.3	20.10.9	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.06.15	1.61.2	20.10.9	专属
Windows_Server-2016-English-Full-ECS_Optimized-2022.01.18	1.57.1	20.10.9	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.12.16	1.57.1	20.10.7	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.11.11	1.57.0	20.10.7	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.09.23	1.55.3	20.10.7	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.08.12	1.55.0	20.10.6	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2021.07.13	1.54.02	20.10.6	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.07.08	1.54.0	20.10.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.06.11	1.53.0	20.10.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.05.21	1.52.2	20.10.4	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.04.14	1.51.0	20.10.0	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.03.11	1.50.2	19.03.14	专属
Windows_Server-2016-English-Full-ECS_Optimized-2021.02.10	1.50.0	19.03.14	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2021.01.13	1.49.0	19.03.14	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.11.18	1.48.0	19.03.13	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.11.06	1.47.0	19.03.11	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.10.14	1.45.0	19.03.12	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.09.09	1.44.3	19.03.11	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.08.12	1.43.0	19.03.11	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.07.15	1.41.1	19.03.5	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2020.06.11	1.40.0	19.03.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.05.14	1.39.0	19.03.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2020.01.15	1.35.0	19.03.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.12.16	1.34.0	19.03.5	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.11.25	1.34.0	19.03.4	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.11.13	1.32.1	19.03.4	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.10.09	1.32.0	19.03.2	专属

经 Amazon ECS 优化的 Windows Server 2016 Full AMI	Amazon ECS 容器代理版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2019.09.11	1.30.0	19.03.1	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.08.16	1.29.1	19.03.1	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.07.19	1.29.0	18.09.8	专属
Windows_Server-2016-English-Full-ECS_Optimized-2019.03.07	1.26.0	18.03.1	专属

使用以下 AWS CLI 经 Amazon ECS 优化的 Windows Server 2016 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

构建您自己的经 Amazon ECS 优化的 Windows AMI

使用 EC2 Image Builder 构建您自定义的经 Amazon ECS 优化的 Windows AMI。这样，您可以轻松地在 Amazon ECS 上使用具有您自己的许可证的 Windows AMI。Amazon ECS 有一个托管 Image Builder 组件，提供了运行 Windows 实例以托管容器所需的系统配置。每个 Amazon ECS 托管组件都包含一个特定的容器代理和 Docker 版本。您可以自定义映像以使用最新的 Amazon ECS 托管组件，或者如果需要较旧的容器代理或 Docker 版本，则可以指定其他组件。

有关使用 EC2 Image Builder 的完整演练，请参阅EC2 Image Builder 入门中的 EC2 Image Builder 用户指南。

使用 EC2 Image Builder 构建您自己的经 Amazon ECS 优化 Windows AMI 时，您可以创建映像配方。您的映像配方必须满足以下要求：

- 源映像应基于 Windows 服务器 2019 Core、Windows Server 2019 Full、Windows Server 2022 Core 或 Windows Server 2022 Full。不支持任何其他 Windows 操作系统，并且可能与该组件不兼容。
- 在指定生成组件，`ecs-optimized-ami-windows` 组件是必需的。推荐 `update-windows` 组件，确保映像包含最新的安全更新。

  要指定其他组件版本，请展开版本控制选项菜单并指定要使用的组件版本。有关更多信息，请参阅列出 `ecs-optimized-ami-windows` 组件版本。

列出 **`ecs-optimized-ami-windows`** 组件版本

创建 EC2 Image Builder 配方并指定 `ecs-optimized-ami-windows` 组件，您可以使用默认选项，也可以指定特定组件版本。要确定可用的组件版本以及组件中包含的 Amazon ECS 容器代理和 Docker 版本，您可以使用 AWS Management Console。

要列出可用的 **`ecs-optimized-ami-windows`** 组件版本

1. 打开位于 https://console.aws.amazon.com/imagebuilder/的 EC2 Image Builder 控制台
2. 在导航栏上，选择构建映像所在的区域。
3. 在导航窗格中的保存的配置菜单下，选择组件。
4. 在组件页面上，在搜索栏中键入 `ecs-optimized-ami-windows`，下拉资质菜单，然后选择 Quick start (Amazon 托管)。
5. 使用说明列确定您映像所需的 Amazon ECS 容器代理和 Docker 版本的组件版本。

Amazon ECS Windows 容器实例管理

当您为 Amazon ECS 工作负载使用 EC2 实例时，您需要负责维护这些实例。

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。

管理程序

- 启动 Amazon ECS Windows 容器实例
- 引导启动 Amazon ECS Windows 容器实例以传递数据
- 为 Amazon ECS Windows 容器实例使用 HTTP 代理
- 配置 Amazon ECS Windows 容器实例以接收竞价型实例通知

启动 Amazon ECS Windows 容器实例

Amazon ECS 容器实例是使用 Amazon EC2 控制台创建的。开始之前，请确保您已完成 设置以使用 Amazon ECS 中的步骤。

有关启动向导的更多信息，请参阅《Amazon EC2 用户指南》中的使用新启动实例向导启动实例。

您可以使用新 Amazon EC2 向导启动实例。您可以使用以下列表中的参数，将未列出的参数作为默认值。以下说明将引导您完成每个参数组。

过程

在开始之前，请完成 设置以使用 Amazon ECS 中的步骤。

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在屏幕顶部的导航栏中，会显示当前 AWS 区域 [例如，美国东部（俄亥俄）]。选择要在其中启动实例的区域。选择该内容是非常重要的，因为可以在区域之间共享某些 Amazon EC2 资源，而无法共享其他资源。
3. 从 Amazon EC2 控制台控制面板中，选择启动实例。

名称和标签

实例名称是一个标签，其中密钥为 Name（名称），而值为您指定的名称。您可以为实例、卷和弹性图形添加标签。对于竞价型实例，您只能标记竞价型实例请求。

指定实例名称和其它标签为可选项。

- 对于 Name（名称），为实例输入一个描述性名称。如果您没有指定名称，则可以通过其 ID 标识实例，该 ID 将在您启动实例时自动生成。
- 要添加其它标签，请选择 Add additional tags（添加其它标签）。选择 Add tag（添加标签），然后输入密钥和值，然后选择要标记的资源类型。为每个要添加的其它标签选择 Add tag（添加标签）。

应用程序和操作系统镜像（亚马逊机器映像）

亚马逊机器映像（AMI）中包含了创建实例所需的信息。例如，AMI 可能包含充当 Web 服务器所需的软件，例如 Apache 和您的网站。

有关最新的经 Amazon ECS 优化的 AMI 及其值，请参阅经 Windows Amazon ECS 优化的 AMI。

使用搜索栏查找由 AWS 发布的合适的经 Amazon ECS 优化的 AMI。

1. 根据您的要求，在搜索栏中输入以下 AMI 之一，然后按 Enter。
   - Windows_Server-2022-English-Full-ECS_Optimized
   - Windows_Server-2022-English-Core-ECS_Optimized
   - Windows_Server-2019-English-Full-ECS_Optimized
   - Windows_Server-2019-English-Core-ECS_Optimized
   - Windows_Server-2016-English-Full-ECS_Optimized
2. 在 Choose an Amazon Machine Image (AMI)（选择 Amazon 机器映像 (AMI)）页面上，选择 Community AMIs（社区 AMI）类别。
3. 从显示的列表中，选择经过 Microsoft 验证且具有最新发布日期的 AMI，然后单击 Select（选择）。

实例类型

实例类型定义了实例的硬件配置和大小。更大的实例类型拥有更多的 CPU 和内存。有关更多信息，请参阅实例类型。

- 对于 Instance type（实例类型），请为实例选择实例类型。

  您选择的实例类型决定了可用于运行您的任务的资源。

密钥对（登录）

为 Key pair name（密钥对名称）选择一个现有密钥对，或选择 Create new key pair（创建新密钥对）来新建一个密钥对。

> **Important**
>
> 如果您选择 Proceed without key pair (Not recommended)（在没有密钥对的情况下继续（不推荐））选项，则将无法连接到此实例，除非您选择配置为允许用户以其它方式登录的 AMI。

Network settings (网络设置)

根据需要配置网络设置。

- 联网平台：选择 Virtual Private Cloud (VPC)（虚拟私有云（VPC）），则在 Network interfaces（网络接口）部分中指定子网。
- VPC：选择要在其中创建安全组的现有 VPC。
- 子网：您可以在与可用区、本地扩展区、Wavelength 区域或 Outpost 关联的子网中启动实例。

  要在可用区中启动实例，请选择要在其中启动实例的子网。要创建新子网，请选择 Create new subnet 转到 Amazon VPC 控制台。完成此操作后，返回到启动实例向导并选择“Refresh”（刷新）图标，以便将您的子网加载到列表中。

  要在本地区域中启动实例，请选择您在本地区域中创建的子网。

  要在 Outpost 中启动实例，请在 VPC 中选择与 Outpost 关联的子网。
- 自动分配公有 IP：如果实例应可从互联网进行访问，请验证 Auto-assign Public IP（自动分配公有 IP）字段设置为 Enable（启用）。如果不是，请将此字段设置为禁用。

  > **Note**
  >
  > 容器实例需要访问才能与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 端点或具有公共 IP 地址的容器实例实现。
  > 有关接口 VPC 端点的更多信息，请参阅 Amazon ECS 接口 VPC 端点 (AWS PrivateLink)
  > 如果您没有配置接口 VPC 端点，并且您的容器实例没有公有 IP 地址，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 为 Amazon ECS Linux 容器实例使用 HTTP 代理。
- Firewall (security groups)（防火墙（安全组））：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。
  - 要选择现有安全组，请选择 Select an existing security group（选择现有安全组），然后选择您在设置以使用 Amazon ECS 中创建的安全组

配置存储

您选择的 AMI 包含一个或多个存储卷，包括根卷。您可以指定要附加到实例的其它卷。

您可以使用 Simple（简单）视图。

- Storage type（存储类型）：为您的容器实例配置存储。

  如果您使用的是经 Amazon ECS 优化的 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 Amazon ECS 优化型 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。

高级详细信息

对于Advanced details (高级详细信息)，请展开该部分以查看字段并为实例指定任何其他参数。

- 购买选项：选择 Request Spot instances（请求竞价型实例）以请求竞价型实例。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。

  .

- IAM 实例配置文件：选择您的容器实例 IAM 角色。其通常被命名为 `ecsInstanceRole`。

  > **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。

- （可选）用户数据：使用用户数据（如 Amazon ECS 容器代理配置 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。

    `EnableTaskIAMRole` 打开任务的任务 IAM 角色功能。

此外，使用 awsvpc 网络模式时，以下选项可用。

- EnableTaskENI：此标志打开任务联网，并且在使用 awsvpc 网络模式时是必需的。
- AwsvpcBlockIMDS：此可选标志阻止在 awsvpc 网络模式下运行的任务容器的 IMDS 访问。
- AwsvpcAdditionalLocalRoutes：此可选标志允许您在任务命名空间中有其他路由。

  将 ip-address 替换为附加路由的IP地址，例如172.31.42.23/32。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -EnableTaskENI -
AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

引导启动 Amazon ECS Windows 容器实例以传递数据

在启动 Amazon EC2 实例时，您可以将用户数据传递到 EC2 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 进程守护程序和 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。

您可以在使用 Amazon EC2 启动向导时传递此用户数据。有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

原定设置 Windows 用户数据

该示例用户数据脚本显示了在使用控制台时 Windows 容器实例接收的默认用户数据。下面的脚本将执行以下操作：

- 将集群名称设置为您输入的名称。
- 设置任务的 IAM 角色。
- 将 json-file 以及 awslogs 设置为可用的日志记录驱动程序。

此外，使用 awsvpc 网络模式时，以下选项可用。

- EnableTaskENI：此标志打开任务联网，并且在使用 awsvpc 网络模式时是必需的。

- `AwsvpcBlockIMDS`：此可选标志阻止在 awsvpc 网络模式下运行的任务容器的 IMDS 访问。
- `AwsvpcAdditionalLocalRoutes`：此可选标志允许您有其他路由。

  将 `ip-address` 替换为附加路由的IP地址，例如172.31.42.23/32。

您可以将该脚本用于自己的容器实例，但前提是它们是从经 Amazon ECS 优化的 Windows Server AMI 启动的。

替换 `-Cluster` *`cluster-name`* 行以指定您自己的集群名称。

```
<powershell>
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]' -EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

对于配置为使用 awslogs 日志记录驱动程序的 Windows 任务，您还必须在容器实例上设置 ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE 环境变量。使用下面的语法。

替换 `-Cluster` *`cluster-name`* 行以指定您自己的集群名称。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Windows 代理安装用户数据

此示例用户数据脚本在使用 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。使用经 Amazon ECS 优化的 Windows AMI 的 Windows 容器更容易上手。有关更多信息，请参阅 创建 Fargate 启动类型的 Amazon ECS 集群。

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>
```

为 Amazon ECS Windows 容器实例使用 HTTP 代理

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互联网网关、NAT 网关或实例访问外部网络，则这非常有用。

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 Amazon ECS 代理用来连接到互联网。例如，在容器实例无法通过 Amazon VPC互联网网关、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the section called “启动容器实例”中的步骤。只需将下面的 PowerShell 脚本复制并粘贴到 User data (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 Amazon EC2 Windows 实例附加配置。

```
<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>
```

配置 Amazon ECS Windows 容器实例以接收竞价型实例通知

当 Spot 价格超过您请求的最高价格或容量不再可用时，Amazon EC2会终止、停止或休眠您的 Spot 实例。Amazon EC2 将提供竞价型实例中断通知，这会在实例中断之前为其提供两分钟的警告。如果在实例上启用了 Amazon ECS Spot 实例耗尽，则 ECS 会收到 Spot 实例中断通知，并将实例置于 DRAINING 状态。

> **Important**
>
> Amazon ECS 监控具有 `terminate` 和 `stop` 实例操作的 Spot 实例中断通知。如果您在请求 Spot 实例或 Spot 队列时指定了 `hibernate` 实例中断行为，则这些实例不支持 Amazon ECS Spot 实例耗尽。

当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。

您可以在启动实例时开启竞价型实例耗尽功能。在启动容器代理之前，您必须设置 ECS_ENABLE_SPOT_INSTANCE_DRAINING 参数。将 *my-cluster* 替换为您集群的名称。

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

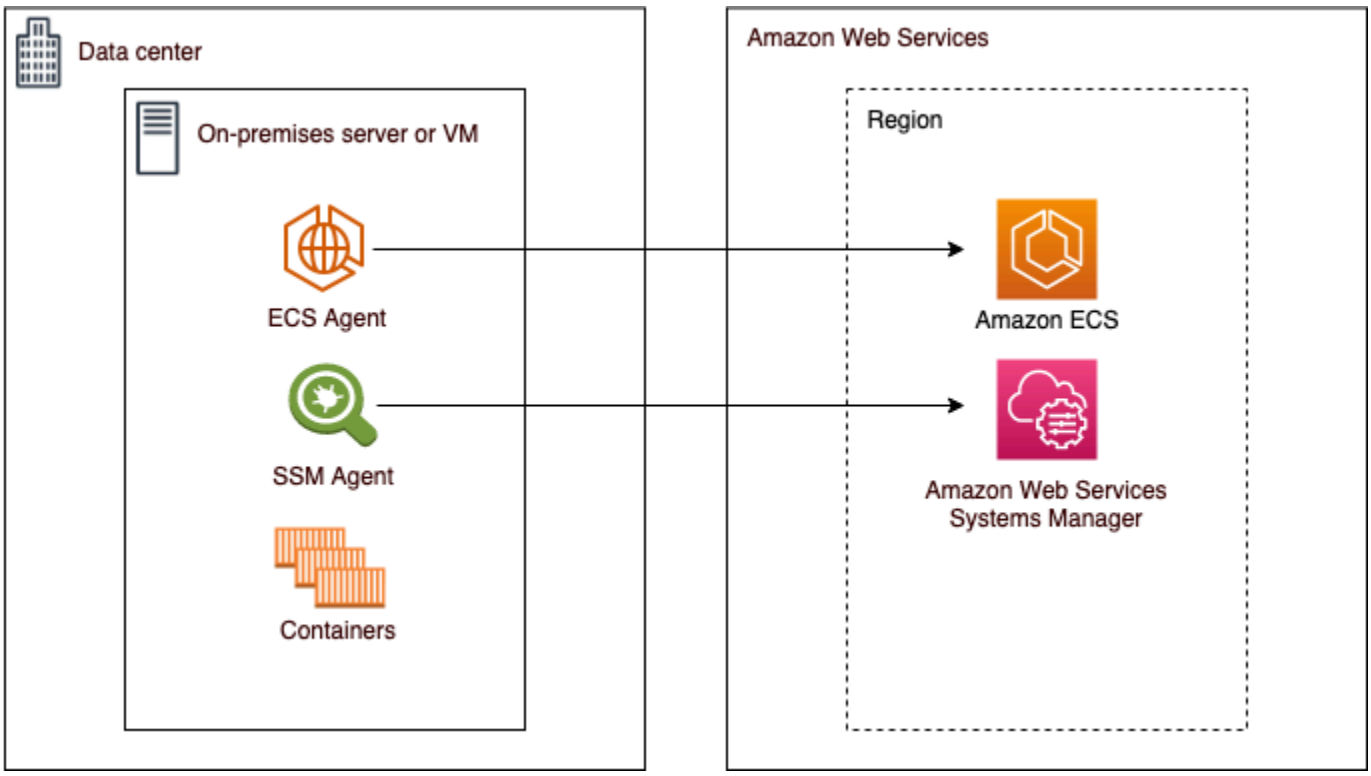
# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

有关更多信息，请参阅 the section called “启动容器实例”。

外部启动类型的 Amazon ECS 集群

Amazon ECS Anywhere 支持向 Amazon ECS 群集注册外部实例，如本地部署服务器或虚拟机（VM）。外部实例针对生成出站流量或流程数据的运行应用程序进行了优化。如果应用程序需要入站流量，则缺乏 Elastic Load Balancing 支持会降低运行这些工作负载的效率。Amazon ECS添加了一个新的 EXTERNAL 启动类型，您可以使用该类型在外部实例上创建服务或运行任务。

下面介绍了 Amazon ECS Anywhere 的高级系统体系结构概述。您的本地服务器同时也安装了 Amazon ECS 代理和 SSM 代理。



支持的操作系统和系统体系结构

以下是受支持的操作系统和系统体系结构的列表。

- Amazon Linux 2
- CentOS 7
- CentOS Stream 8
- RHEL 7、RHEL 8 — Docker 或 RHEL 的开放程序包存储库都不支持在 RHEL 上本地安装 Docker。在运行本文档中描述的安装脚本之前，您必须确保已安装 Docker。
- Fedora 32、Fedora 33

- openSUSE Tumbleweed
- Ubuntu 18、Ubuntu 20、Ubuntu 22
- Debian 10

> **Important**
>
> Debian 9 长期支持（LTS 支持）已于 2022 年 6 月 30 日结束，并且不再受 Amazon ECS Anywhere 支持。

- Debian 11
- Debian 12 – Debian 12 目前不支持 NVIDIA 容器工具包。您将无法在 Debian 12 实例上运行 GPU。
- SUSE Enterprise Server 15
- 支持 x86_64 和 ARM64 CPU 体系结构。
- 支持以下 Windows 操作系统版本：
  - Windows Server 2022
  - Windows Server 2019
  - Windows Server 2016
  - Windows Server 20H2

注意事项

在开始使用外部实例之前，请注意以下注意事项。

- 您可以一次向一个集群注册一个外部实例。有关如何向其他集群注册外部实例的说明，请参阅 注销 Amazon ECS 外部实例。
- 您的外部实例需要 IAM 角色，允许它们与 AWS API通信。有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色。
- 您的外部实例不应具有本地定义的预配置实例凭据链，因为这会干扰注册脚本。
- 要将容器日志发送到 CloudWatch Logs，请确保在任务定义中创建并指定任务执行 IAM 角色。
- 当外部实例注册到集群时，`ecs.capability.external` 属性与实例相关联。此属性将实例标识为外部实例。可以将自定义属性添加到外部实例中，用作任务放置约束。有关更多信息，请参阅 自定义属性。
- 您可以将资源标签添加到外部实例。有关更多信息，请参阅 外部容器实例。
- ECS Exec 在外部实例上受支持。有关更多信息，请参阅 使用 ECS Exec 监控 Amazon ECS 容器。

- 以下是特定于与外部实例联网的其他注意事项。有关更多信息，请参阅 联网 。
  - 不支持服务负载平衡。
  - 不支持服务发现。
  - 在外部实例上运行的任务必须使用 bridge、host 或 none 网络模式。不支持 awsvpc 网络模式。
  - 每个 AWS 区域都有 Amazon ECS 服务域。必须允许这些服务域向外部实例发送流量。
  - 安装在外部实例上的 SSM Agent 维护使用硬件指纹每 30 分钟轮换一次的 IAM 凭证。如果您的外部实例与 AWS 断开连接，SSM Agent 会在重新建立连接后自动刷新凭证。有关更多信息，请参阅 AWS Systems Manager 用户指南中的使用硬件指纹验证本地部署服务器和虚拟机。
- 不支持 UpdateContainerAgent API。有关如何更新外部实例上的 SSM Agent 或 Amazon ECS 代理的说明，请参阅 更新外部实例上的 AWS Systems Manager 代理和 Amazon ECS 容器代理。
- 不支持 Amazon ECS 容量提供程序。要在外部实例上创建服务或运行独立任务，请使用 EXTERNAL 启动任务。
- 不支持 SELinux。
- 不支持使用 Amazon EFS 卷或指定 EFSVolumeConfiguration。
- 不支持与 App Mesh 集成。
- 如果您使用控制台创建外部实例任务定义，则必须使用控制台 JSON 编辑器来创建任务定义。
- 在 Windows 上运行 ECS Anywhere 时，必须在本地基础设施上使用自己的 Windows 许可证。
- 当您使用非 Amazon ECS 优化的 AMI 时，请在外部容器实例上运行以下命令来配置规则，以便将 IAM 角色用于任务。有关更多信息，请参阅 外部实例附加配置。

```
$ sysctl -w net.ipv4.conf.all.route_localnet=1
$ 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
```

联网

Amazon ECS 外部实例针对运行生成出站流量或处理数据的应用程序进行了优化。如果您的应用程序需要入站流量，例如web服务，则缺乏 Elastic Load Balancing 支持会降低运行这些工作负载的效率，因为不支持将这些工作负载置于负载平衡器之后。

以下是特定于与外部实例联网的其他注意事项。

- 不支持服务负载平衡。
- 不支持服务发现。
- 在外部实例上运行的 Linux 任务必须使用 `bridge`、`host` 或 `none` 网络模式。不支持 `awsvpc` 网络模式。

  有关各网络模式的更多信息，请参阅 Amazon ECS 最佳实践指南中的选择网络模式。
- 在外部实例上运行的 Windows 任务必须使用 `default` 网络模式。
- 每个区域都有 Amazon ECS 服务域，必须允许其向外部实例发送流量。
- 安装在外部实例上的 SSM Agent 维护使用硬件指纹每 30 分钟轮换一次的 IAM 凭证。如果您的外部实例与 AWS 断开连接，SSM Agent 会在重新建立连接后自动刷新凭证。有关更多信息，请参阅 AWS Systems Manager 用户指南中的使用硬件指纹验证本地部署服务器和虚拟机。

以下域用于 Amazon ECS 服务和安装在您的外部实例上的 Amazon ECS 代理之间的通信。确保允许流量并且 DNS 解析工作正常。对于每个端点，*##*表示 Amazon ECS 支持的 AWS 区域的区域标识符，例如美国东部（俄亥俄州）区域 `us-east-2`。应允许使用所有区域的端点。对于 `ecs-a` 和 `ecs-t` 端点，应包含星号（例如，`ecs-a-*`）。

- `ecs-a-*.`*`region`*`.amazonaws.com`— 托管任务时使用此端点。
- `ecs-t-*.`*`region`*`.amazonaws.com`— 此端点用于托管任务和容器指标。
- `ecs.`*`region`*`.amazonaws.com`— 这是 Amazon ECS 的服务端点。
- `ssm.`*`region`*`.amazonaws.com` — 这是 AWS Systems Manager 的服务端点。
- `ec2messages.`*`region`*`.amazonaws.com` — 这是 AWS Systems Manager 用于在云中的 Systems Manager 代理和 Systems Manager 服务之间进行通信的服务端点。
- `ssmmessages.`*`region`*`.amazonaws.com` — 这是使用云中的 Session Manager 服务创建和删除会话通道所需的服务端点。
- 如果您的任务需要与任何其他 AWS 服务通信，请确保允许这些服务端点可用。示例应用程序包括使用 Amazon ECR 提取容器镜像或使用 CloudWatch Logs 的 CloudWatch。有关更多信息，请参阅 AWS 一般引用指南中的服务端点。

Amazon FSx for Windows File Server 与 ECS Anywhere

要将 Amazon FSx for Windows File Server 与 Amazon ECS 外部实例结合使用，您必须在您的本地数据中心与 AWS Cloud 之间建立连接。有关将您的网络连接到 VPC 的选项的信息，请参阅 Amazon Virtual Private Cloud 连接性选项。

gMSA 与 ECS Anywhere

ECS Anywhere 支持以下使用案例。

- Active Directory 位于 AWS Cloud 中 - 对于此配置，您将使用 AWS Direct Connect 连接在您的本地网络和 AWS Cloud 之间创建连接。有关如何创建连接的信息，请参阅 Amazon Virtual Private Cloud 连接性选项。您可以在 AWS Cloud 中创建 Active Directory。有关如何开始使用 AWS Directory Service 的信息，请参阅 AWS Directory Service 管理指南中的设置 AWS Directory Service。然后，您可以使用 AWS Direct Connect 连接将外部实例加入域。有关将 gMSA 与 Amazon ECS 结合使用的信息，请参阅 the section called “了解如何将 gMSA 用于 EC2 Windows 容器”。
- Active Directory 位于本地数据中心。- 对于此配置，您将您的外部实例加入到本地 Active Directory。然后，在运行 Amazon ECS 任务时，您可以使用本地可用的凭证。

创建外部启动类型的 Amazon ECS 集群

您可以使用 Amazon ECS 控制台创建 Amazon ECS 集群。开始之前，请确保您已完成 设置以使用 Amazon ECS 中的步骤，然后分配相应的 IAM 权限。有关更多信息，请参阅 the section called “Amazon ECS 集群示例”。Amazon ECS 控制台提供了一种简单的方法，通过创建 AWS CloudFormation 堆栈来创建 Amazon ECS 集群所需的资源。

为了使集群创建过程尽可能简单，控制台对许多选项进行了原定设置选择，我们将在下面介绍这些选项。控制台中的大多数部分还提供了帮助面板，以提供进一步的上下文。

- 在 AWS Cloud Map 中创建与集群名称相同的默认命名空间。命名空间允许您在集群中创建的服务无需额外配置即可连接到命名空间中的其他服务。

  有关更多信息，请参阅 互连 Amazon ECS 服务。

您可以修改以下选项：

- 更改与集群关联的默认命名空间。

  命名空间允许您在集群中创建的服务无需额外配置即可连接到命名空间中的其他服务。默认命名空间与集群名称相同。有关更多信息，请参阅 互连 Amazon ECS 服务。
- 为外部实例配置集群
- 开启 Container Insights。

CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。Container Insights 还提供诊断信息（如容器重新启动失败），您可以用该信息查明问题并快速解决问题。有关更多信息，请参阅 the section called “使用 Container Insights 监控 Amazon ECS 容器”。

- 添加标签以帮助您识别集群。

要创建新集群（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters（集群）页面上，选择 Create cluster（创建集群）。
5. 在集群配置下，配置以下内容：
   - 对于集群名称，输入唯一的名称。

     该名称最多可以包含 255 个字母（大小写字母）、数字和连字符。
   - （可选）要使用于 Service Connect 的命名空间与集群名称不同，请在命名空间中输入一个唯一的名称。
6. 展开基础设施，选择 AWS Fargate（无服务器）。
7. （可选）要打开 Container Insights，请展开 Monitoring（监控），然后打开 Use Container Insights（使用 Container Insights）。
8. （可选）为了帮助识别您的集群，请展开 Tags（标签），然后配置您的标签。

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。
9. 选择创建。

后续步骤

您必须向集群注册这些实例。有关更多信息，请参阅 将外部实例注册到 Amazon ECS 集群。

创建集群后，您可以针对应用程序创建任务定义，然后将其作为独立任务或服务的一部分运行。有关更多信息，请参阅下列内容：

- Amazon ECS 任务定义
- 将应用程序作为 Amazon ECS 任务运行
- 使用控制台创建 Amazon ECS 服务

将外部实例注册到 Amazon ECS 集群

对于您向 Amazon ECS 集群注册的每个外部实例，必须安装 SSM Agent、Amazon ECS 容器代理和 Docker。要将外部实例注册到 Amazon ECS 集群，必须首先将其注册为 AWS Systems Manager 托管实例。您可以在 Amazon ECS 控制台上单击几下即可创建安装脚本。安装脚本包括一个 Systems Manager 激活密钥和命令，用于安装每个所需代理和 Docker。必须在本地部署服务器或 VM 上运行安装脚本，才能完成安装和注册步骤。

> **Note**
>
> 在将 Linux 外部实例注册到集群之前，请在您的外部实例上创建 `/etc/ecs/ecs.config` 文件，然后添加所需的任何容器代理配置参数。在将外部实例注册到集群后，您无法执行此操作。有关更多信息，请参阅 Amazon ECS 容器代理配置。

AWS Management Console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在集群页面上，选择要将外部实例注册到的集群。
5. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。
6. 在 Register external instances（注册外部实例）页面上，完成以下步骤。
   a. 对于激活密钥持续时间（以天为单位），输入激活密钥保持活动状态的天数。在您输入的天数过后，注册外部实例时，密钥将不再有效。
   b. 对于实例的数量，输入要使用激活密钥注册到集群的外部实例数量。

c. 对于实例角色，选择要与外部实例关联的 IAM 角色。如果尚未创建角色，请选择创建新角色让 Amazon ECS 代表您创建角色。有关外部实例需要哪些 IAM 权限的更多信息，请参阅 Amazon ECS Anywhere IAM 角色。

d. 复制注册命令。应在要注册到群集的每个外部实例上运行此命令。

> **Important**
>
> 脚本的 bash 部分必须以根用户的身份运行。如果未以根用户身份运行命令，则返回错误。

e. 选择关闭。

AWS CLI for Linux operating systems

1. 创建 Systems Manager 激活对。这用于 Systems Manager 托管实例激活。输出包括 `ActivationId` 和 `ActivationCode`。您将在后面的步骤中用到它。确保您指定了创建的 ECS Anywhere IAM 角色。有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色。

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. 在本地部署服务器或虚拟机 (VM) 上，下载安装脚本。

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"
```

3. （可选）在本地部署服务器或虚拟机 (VM) 上，使用脚本签名文件验证安装脚本。

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。

```
gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
```

c. 下载安装脚本签名。签名是存储在扩展名为 `.asc` 的文件中的 ascii 分离 PGP 签名。

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh.asc" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh.asc"
```

d. 使用密钥验证安装脚本文件。

```
gpg --verify /tmp/ecs-anywhere-install.sh.asc /tmp/ecs-anywhere-install.sh
```

预期的输出如下所示：

```
gpg: Signature made Tue 25 May 2021 07:16:29 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
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
```

4. 在本地部署服务器或虚拟机 (VM) 上，运行安装脚本。从第一步指定群集名称、区域和 Systems Manager 激活 ID 和激活代码。

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE
```

对于为 GPU 工作负载安装 NVIDIA 驱动程序的本地服务器或虚拟机（VM），您必须添加 `--enable-gpu` 标记到安装脚本。指定此标记后，安装脚本将验证 NVIDIA 驱动程序是否正在运行，然后添加运行 Amazon ECS 任务所需的配置变量。有关在任务定义中运行 GPU 工作负载和指定 GPU 要求的更多信息，请参阅 在 Amazon ECS 任务定义中指定 GPU 数。

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE \
    --enable-gpu
```

使用以下步骤将现有外部实例注册到其他集群。

将现有外部实例注册到其他集群

1. 停止 Amazon ECS 容器代理。

```
sudo systemctl stop ecs.service
```

2. 编辑 /etc/ecs/ecs.config 文件，并在 ECS_CLUSTER 行中确保集群名称与要向其注册外部实例的集群名称相匹配。
3. 删除现有的 Amazon ECS 代理数据。

```
sudo rm /var/lib/ecs/data/agent.db
```

4. 开始 Amazon ECS 容器代理

```
sudo systemctl start ecs.service
```

AWS CLI for Windows operating systems

1. 创建 Systems Manager 激活对。这用于 Systems Manager 托管实例激活。输出包括 ActivationId 和 ActivationCode。您将在后面的步骤中用到它。确保您指定了创建的 ECS Anywhere IAM 角色。有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色。

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. 在本地部署服务器或虚拟机 (VM) 上，下载安装脚本。

```
Invoke-RestMethod -URI "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install.ps1" -OutFile “ecs-anywhere-install.ps1”
```

3. （可选）Powershell 脚本由 Amazon 签名，因此，Windows 会自动对同一脚本执行证书验证。您无需执行任何手动验证。

   要手动验证证书，请右键单击该文件，导航到属性，然后使用 Digital Signatures（数字签名）选项卡获取更多详细信息。

   仅当主机在证书存储中有证书时，此选项才可用。

   此验证应返回类似于以下内容的信息：

```
# Verification (PowerShell)
Get-AuthenticodeSignature -FilePath .\ecs-anywhere-install.ps1

SignerCertificate                         Status      Path
```

```
-----------------                              ------      ----
EXAMPLECERTIFICATE                             Valid       ecs-anywhere-install.ps1

...

Subject               : CN="Amazon Web Services, Inc.",...

----
```

4. 在本地部署服务器或虚拟机 (VM) 上，运行安装脚本。从第一步指定群集名称、区域和 Systems Manager 激活 ID 和激活代码。

```
.\ecs-anywhere-install.ps1 -Region $Region -Cluster $Cluster -
ActivationID $ActivationID -ActivationCode $ActivationCode
```

5. 验证 Amazon ECS 容器代理正在运行。

```
Get-Service AmazonECS

Status   Name               DisplayName
------   ----               -----------
Running  AmazonECS          Amazon ECS
```

使用以下步骤将现有外部实例注册到其他集群。

将现有外部实例注册到其他集群

1. 停止 Amazon ECS 容器代理。

```
Stop-Service AmazonECS
```

2. 修改 ECS_CLUSTER 参数，以便集群名称与要向其注册外部实例的集群名称相匹配。

```
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", $ECSCluster,
 [System.EnvironmentVariableTarget]::Machine)
```

3. 删除现有的 Amazon ECS 代理数据。

```
Remove-Item -Recurse -Force $env:ProgramData\Amazon\ECS\data\*
```

4. 开始 Amazon ECS 容器代理

```
Start-Service AmazonECS
```

在运行安装脚本以完成外部实例注册过程之前，可以使用 AWS CLI 来创建 Systems Manager 激活。

注销 Amazon ECS 外部实例

我们建议您在完成使用实例后，从 Amazon ECS 和 AWS Systems Manager 注销该实例。注销后，容器实例再也不能接受新任务。

如果注销时容器实例上有正在运行的任务，这些任务将保持运行，直到通过其他方式停止为止。但是，这些任务不再由 Amazon ECS 监控或说明。如果外部实例上的这些任务是 Amazon ECS 服务的一部分，那么服务调度器将在不同的实例上启动该任务的另一个副本（如有可能）。

注销实例之后，请清理实例上剩余的 AWS 资源。然后，您可以将其注册到新的集群。

过程

AWS Management Console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择您的外部实例将注册的区域。
3. 在导航窗格中，选择 集群 并选择托管外部实例的集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。
5. 在 Container instances（容器实例）下，选择要注销的外部实例 ID。您将重新定向到容器实例详细信息页面。
6. 在 Container Instance : ***id*** 页面上，选择 Deregister。
7. 查看注销消息。选择注销 AWS Systems Manager 以同时将外部实例注销为 Systems Manager 托管实例。选择注销。

> **Note**
>
> 您可以在 Systems Manager 控制台中将外部实例注销为 Systems Manager 托管实例。有关说明，请参阅 AWS Systems Manager 用户指南中的注销托管实例。

8. 注销实例之后，清理本地服务器或 VM 上的 AWS 资源。

操作系统	步骤	

Linux	a. 停止实例上的 Amazon ECS 容器代理和 SSM Agent 服务。
	`sudo systemctl stop ecs amazon-ssm-agent`
	b. 删除 Amazon ECS 和 Systems Manager 程序包。
	对于 CentOS 7、CentOS 8 和 RHEL 7
	`sudo yum remove -y amazon-ecs-init amazon-ssm-agent`
	对于 SUSE Linux Enterprise Server 15
	`sudo zypper remove -y amazon-ecs-init amazon-ssm-agent`
	对于 Ubuntu 和 Debian
	`sudo apt remove -y amazon-ecs-init amazon-ssm-agent`
	c. 移除剩余的目录。
	`sudo rm -rf /var/lib/ecs /etc/ecs /var/lib/amazon/ss`	

操作系统	步骤
	`m /var/log/ecs / var/log/amazon/ssm`

Windows	a. 停止实例上的 Amazon ECS 容器代理和 SSM Agent 服务。
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. 删除 Amazon ECS 程序包。
	`.\ecs-anywhere-install.ps1 -Uninstall`

AWS CLI

1. 您需要实例 ID 和容器实例 ARN，才能注销容器实例。如果没有这些值，则请运行以下命令

   请运行以下命令以获取实例 ID。

   您可以使用实例 ID（`instanceID`）来获取容器实例 ARN（`containerInstanceARN`）。

   ```
   instanceId=$(aws ssm describe-instance-information --region "{{ region }}" |
    jq ".InstanceInformationList[] |select(.IPAddress==\"{{ IPv4 Address }}\")
    | .InstanceId" | tr -d'"'
   ```

   运行以下命令。

   您可以在命令中使用 `containerInstanceArn` 作为参数来注销实例（`deregister-container-instance`）。

```
instances=$(aws ecs list-container-instances --cluster "{{ cluster }}" --region
 "{{ region }}" | jq -c '.containerInstanceArns')
containerInstanceArn=$(aws ecs describe-container-instances --cluster
 "{{ cluster }}" --region "{{ region }}" --container-instances $instances
 | jq ".containerInstances[] | select(.ec2InstanceId==\"{{ instanceId }}\")
 | .containerInstanceArn" | tr -d '"')
```

2. 运行以下命令以耗尽实例。

```
aws ecs update-container-instances-state --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instances "{{ containerInstanceArn }}" --status
 DRAINING
```

3. 容器实例耗尽后，运行以下命令注销该实例。

```
aws ecs deregister-container-instance --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instance "{{ containerInstanceArn }}"
```

4. 使用以下命令从 SSM 中移除容器实例。

```
aws ssm deregister-managed-instance --region "{{ region }}" --instance-id
 "{{ instanceId }}"
```

5. 注销实例之后，清理本地服务器或 VM 上的 AWS 资源。

操作系统	步骤	
Linux	a. 停止实例上的 Amazon ECS 容器代理和 SSM Agent 服务。 `sudo systemctl stop ecs amazon-ssm-agent` b. 删除 Amazon ECS 和 Systems Manager 程序包。	

操作系统	步骤

	`sudo (yum/apt/zypper) remove amazon-ecs-init amazon-ssm-agent`
	c. 移除剩余的目录。
	`sudo rm -rf /var/lib/ecs /etc/ecs / var/lib/amazon/ssm /var/log/ecs / var/log/amazon/ssm`

Windows	a. 停止实例上的 Amazon ECS 容器代理和 SSM Agent 服务。
	`Stop-Service AmazonECS`
	`Stop-Service AmazonSSMAgent`
	b. 删除 Amazon ECS 程序包。
	`.\ecs-anywhere-install.ps1 -Uninstall`

更新外部实例上的 AWS Systems Manager 代理和 Amazon ECS 容器代理

运行 Amazon ECS 工作负载时，您的本地部署服务器或 VM 必须同时运行 AWS Systems Manager 代理 (SSM Agent) 和 Amazon ECS 容器代理。AWS 在添加或更新任何功能时发布这些代理的新版本。如果您的外部实例正在使用任一代理的早期版本，则可以使用以下过程更新。

更新外部实例上的 SSM Agent

AWS Systems Manager 建议您自动完成更新实例上的 SSM Agent 的过程。它们提供了多种自动更新的方法。有关更多信息，请参阅 AWS Systems Manager 用户指南中的自动更新 SSM Agent。

更新外部实例上的 Amazon ECS 代理

在您的外部实例上，Amazon ECS 容器代理将通过升级 `ecs-init` 程序包更新。更新 Amazon ECS 代理不会中断正在运行的任务或服务。Amazon ECS在每个区域的 Amazon S3 存储桶中提供 `ecs-init` 程序包和签名文件。从 `ecs-init`版本 `1.52.1-1` 开始，Amazon ECS 提供单独的 `ecs-init` 程序包，具体取决于您的外部实例使用的操作系统和系统体系结构。

可以使用下表确定 `ecs-init` 程序包，您应该根据外部实例使用的操作系统和系统体系结构下载该程序包。

> **Note**
>
> 您可以使用以下命令确定外部实例使用的操作系统和系统体系结构。
>
> ```
> cat /etc/os-release
> uname -m
> ```

操作系统（体系结构）	ecs-init 程序包
CentOS 7 (x86_64) CentOS 8 (x86_64) SUSE Enterprise Server 15 (x86_64) RHEL 7 (x86_64) RHEL 8 (x86_64)	`amazon-ecs-init-latest.x86_64.rpm`
CentOS 7 (aarch64) CentOS 8 (aarch64) RHEL 7 (aarch64)	`amazon-ecs-init-latest.aarch64.rpm`

操作系统（体系结构）	ecs-init 程序包
Debian 9 (x86_64) Debian 10 (x86_64) Debian 11 (x86_64) Debian 12 (x86_64) Ubuntu 18 (x86_64) Ubuntu 20 (x86_64)	amazon-ecs-init-latest.amd64.deb
Debian 9 (aarch64) Debian 10 (aarch64) Debian 11 (aarch64) Debian 12 (aarch64) Ubuntu 18 (aarch64) Ubuntu 20 (aarch64)	amazon-ecs-init-latest.arm64.deb

请按照以下步骤更新 Amazon ECS 代理。

要更新 Amazon ECS 代理

1. 确认您正在运行的 Amazon ECS 代理版本。

   ```
   curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
   ```

2. 下载适用于您的操作系统和系统体系结构的ecs-init 程序包。Amazon ECS在每个区域的 Amazon S3 存储桶中提供 ecs-init 程序包文件。确保将命令中的*<region>*标识符替换为地理位置最接近的区域名称（例如，us-west-2）。

   amazon-ecs-init-latest.x86_64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.x86_64.rpm
```

amazon-ecs-init-latest.aarch64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.aarch64.rpm
```

amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.amd64.deb
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-
agent-<region>/amazon-ecs-init-latest.arm64.deb
```

3. （可选）用 PGP 签名验证 `ecs-init` 程序包文件。

   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。

   ```
   gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
   ```

   c. 下载 `ecs-init` 程序包签名文件。签名是存储在扩展名为 `.asc` 的文件中的 ASCII 分离 PGP 签名。Amazon ECS在每个区域的 Amazon S3 存储桶中提供签名文件。确保将命令中的*<region>*标识符替换为地理位置最接近的区域名称（例如，`us-west-2`）。

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-
   agent-<region>/amazon-ecs-init-latest.x86_64.rpm.asc
   ```

   amazon-ecs-init-latest.aarch64.rpm

```
curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm.asc
```

amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb.asc
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb.asc
```

d. 用密钥验证 `ecs-init` 程序包文件。

对于 **rpm** 程序包

```
gpg --verify amazon-ecs-init.rpm.asc ./amazon-ecs-init.rpm
```

对于 **deb** 程序包

```
gpg --verify amazon-ecs-init.deb.asc ./amazon-ecs-init.deb
```

预期的输出如下所示：

```
gpg: Signature made Fri 14 May 2021 09:31:36 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
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
```

4. 安装 `ecs-init` 软件包。

对于 CentOS 7、CentOS 8 和 RHEL 7 上的 **rpm** 程序包

```
sudo yum install -y ./amazon-ecs-init.rpm
```

对于SUSE 企业服务器 15 上的 **rpm** 程序包

```
sudo zypper install -y --allow-unsigned-rpm ./amazon-ecs-init.rpm
```

对于 **deb** 程序包

```
sudo dpkg -i ./amazon-ecs-init.deb
```

5. 重新启动 ecs 服务。

```
sudo systemctl restart ecs
```

6. 验证 Amazon ECS 代理版本是否已更新。

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

更新 Amazon ECS 集群

您可以对以下集群属性进行修改：

- 设置默认的容量提供程序

  每个集群可以有一个或多个容量提供程序和一个可选的容量提供程序策略。容量提供程序策略确定任务在集群的容量提供程序之间的分布方式。运行独立任务或创建服务时，您可以使用集群的默认容量提供程序策略，也可以使用覆盖默认策略的容量提供程序策略。
- 开启 Container Insights。

  CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。Container Insights 还提供诊断信息（如容器重新启动失败），您可以用该信息查明问题并快速解决问题。有关更多信息，请参阅 the section called “使用 Container Insights 监控 Amazon ECS 容器”。
- 添加标签以帮助您识别集群。

过程

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。

3. 在 Clusters（集群）页面上，选择集群。
4. 在集群：**##**页面上，选择更新集群。
5. 要设置默认的容量提供程序，请在默认容量提供程序策略下，选择添加更多。

   a. 对于容量提供程序，请选择容量提供程序。

   b. （可选）对于基本，输入在容量提供程序上运行的最小任务数。

      您只能为一个容量提供程序设置一个基本值。

   c. （可选）对于权重，请输入使用指定容量提供程序的已启动任务总数的相对百分比。

   d. （可选）对任何其他容量提供程序重复上述步骤。
6. 要打开或关闭 Container Insights，请展开监控，然后打开使用 Container Insights。
7. 为了帮助识别您的集群，请展开标签，然后配置您的标签。

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：

   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。

   [删除标签] 选择标签的“键”和“值”右侧的Remove（删除）。
8. 选择更新。

删除 Amazon ECS 集群

使用完集群后，可以将其删除。删除集群后，它会转换到 `INACTIVE` 状态。具有 `INACTIVE` 状态的集群可能会在一段时间内在您的账户中保持可被发现。但是，此行为可能会在将来发生变化，因此您不应该依赖于持续存在的 `INACTIVE` 集群。

在删除集群前，您必须先执行以下操作：

- 删除集群中的所有服务。有关更多信息，请参阅 the section called “删除服务”。
- 停止当前正在运行的所有任务。有关更多信息，请参阅 the section called “停止任务”。
- 在集群中注销所有已注册的容器实例。有关更多信息，请参阅 the section called “注销容器实例”。
- 删除 命名空间。有关更多信息，请参阅 AWS Cloud Map开发人员指南中的删除命名空间。

过程

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择集群。
4. 在 Clusters (集群) 页面上，选择要删除的集群。
5. 在页面的右上角，选择删除集群。

   当您没有删除与集群关联的所有资源时，将显示一条消息。
6. 在确认框中，输入 delete ***cluster name***。

创建 Amazon ECS 的容量提供程序

集群创建完成后，您可以为 EC2 启动类型创建新容量提供程序（自动扩缩组）。

在创建容量提供程序之前，您需要先创建自动扩缩组。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的自动扩缩组。

为集群创建容量提供程序（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施），然后选择 Create（创建）。
5. 在 Create capacity providers（创建容量提供程序）页面上，配置以下选项。

   a. 在 Basic details（基本详细信息）下，对于 Capacity provider name（容量提供程序名称），输入唯一的容量提供程序名称。

   b. 在 Auto Scaling group（自动扩缩组）下，对于 Use an existing Auto Scaling group（使用现有的自动扩缩组），选择自动扩缩组。

   c. （可选）要配置扩缩策略，请在 Scaling policies（扩展策略）下配置以下选项。

      - 要让 Amazon ECS 管理横向缩减和横向扩展操作，请选择 Turn on managed scaling（开启托管扩展）。

- 要防止具有正在运行的 Amazon ECS 任务的 EC2 实例被终止，请选择 Turn on scaling protection（开启扩展保护）。
- 对于 Set target capacity（设置目标容量），输入在 Amazon ECS 托管式目标跟踪扩展策略中使用的 CloudWatch 指标的目标值。

6. 选择创建。

更新 Amazon ECS 容量提供程序

使用 自动扩缩组作为容量提供程序时，可以修改该组的扩展策略。

更新集群容量提供程序（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施），然后选择 Update（更新）。
5. 在 Create capacity providers（创建容量提供程序）页面上，配置以下选项。
   - 在自动扩缩组的扩缩策略下，配置以下选项。
     - 要让 Amazon ECS 管理横向缩减和横向扩展操作，请选择 Turn on managed scaling（开启托管扩展）。
     - 要防止具有正在运行的 Amazon ECS 任务的 EC2 实例被终止，请选择开启扩展保护。
     - 对于 Set target capacity（设置目标容量），输入在 Amazon ECS 托管式目标跟踪扩展策略中使用的 CloudWatch 指标的目标值。
6. 选择更新。

删除 Amazon ECS 容量提供程序

使用完自动扩缩组容量提供程序后，可以将其删除。删除组后，自动扩缩组容量提供程序将转换为 `INACTIVE` 状态。具有 `INACTIVE` 状态的容量提供程序可能会在一段时间内在您的账户中保持可被发现。但是，此行为可能会在将来发生变化，因此您不应该依赖于持续存在的 `INACTIVE` 容量提供程序。在删除自动扩缩组容量提供程序之前，必须从所有服务的容量提供程序策略中删除该容量提供程序。您可以使用 `UpdateService` API 或 Amazon ECS 控制台中的更新服务工作流，从服务的容量

提供程序策略中删除容量提供程序。使用强制新部署选项可确保使用该容量提供程序提供的 Amazon EC2 实例容量的所有任务均转换为使用剩余容量提供程序提供的容量。

要为集群删除容量提供程序（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 在 Cluster : ***name***（集群：名称）页面上，依次选择 Infrastructure（基础设施）、自动扩缩组，然后选择 Delete（删除）。
5. 在确认框中，输入 delete *******。
6. 选择 删除。

注销 Amazon ECS 容器实例

> **Important**
>
> 本主题仅适用于在 Amazon EC2 中创建的容器实例。有关注销外部实例的更多信息，请参阅注销 Amazon ECS 外部实例。

在您使用完 Amazon EC2 支持的容器实例后，可从您的集群中注销它。取消注册后，容器实例再也不能接受新任务。

如果在您注销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，这意味着它们不再由 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)。向应用程序负载均衡器目标组注册的孤立服务任务中的所有容器都将注销。它们将根据负载均衡器或目标组上的设置开始连接耗尽。如果孤立任务正在使用 `awsvpc` 网络模式时，它们的弹性网络接口将被删除。

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。

注销容器实例时，请注意以下注意事项。

- 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注销，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新的启动

新容器实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的终止实例和 启动 Amazon ECS Linux 容器实例。

- 如果由 自动扩缩组或 AWS CloudFormation 堆栈托管容器实例，通过更新 自动扩缩组或 AWS CloudFormation 堆栈终止实例。否则，自动扩缩组或 AWS CloudFormation 将在终止之后创建新实例。
- 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注销实例。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注销。
- 注销容器实例将从集群中删除该实例，但不会终止 Amazon EC2 实例。如果使用完该实例，请务必终止它以停止计费。有关更多信息，请参阅《Amazon EC2 用户指南》中的 终止实例。

过程

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择您的外部实例将注册的区域。
3. 在导航窗格中，选择 Clusters（集群）并选择托管实例的集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。
5. 在 Container instances（容器实例）下，选择要注销的实例 ID。您将重新定向到容器实例详细信息页面。
6. 在 Container Instance : ***id*** 页面上，选择 Deregister。
7. 在确认屏幕上，选择注销。
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》中的终止实例。

耗尽 Amazon ECS 容器实例

有时可能需要从集群中移除容器实例；例如，要执行系统更新，或缩减集群容量。Amazon ECS 能够将容器实例转换为 `DRAINING` 状态。这称为容器实例耗尽。当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。

服务的耗尽行为

作为服务一部分的任何处于 `PENDING` 状态的任务都将立即停止。如果集群中存在可用容器实例容量，则服务计划程序将启动替换任务。如果容器实例容量不足，将发送一条指出问题的服务事件消息。

作为容器实例上处于状态 RUNNING 的服务的一部分的任务将转换为状态 STOPPED。服务计划程序尝试根据服务的部署类型和部署配置参数、minimumHealthyPercent 和 maximumPercent 替换任务。有关更多信息，请参阅Amazon ECS 服务 和Amazon ECS 服务定义参数。

- 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。

  > **Important**
  >
  > 如果您使用的竞价型实例和 minimumHealthyPercent 大于或等于 100%，则服务将没有足够的时间在竞价型实例终止之前替换任务。

- 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。

  > **Important**
  >
  > 如果 minimumHealthyPercent 和 maximumPercent 都是 100%，那么服务无法删除现有任务，也无法启动替换任务。这可以防止成功耗尽容器实例并防止进行新的部署。

独立任务的耗尽行为

处于 PENDING 或 RUNNING 状态的任何独立任务不受影响；您必须等待它们自行停止或手动停止。容器实例将保留在 DRAINING 状态。

当实例上运行的所有任务都转换为 STOPPED 状态时，容器实例已完成排空。在再次激活或删除容器实例之前，容器实例将保持为状态 DRAINING。您可以使用带有 containerInstance 参数的 ListTasks 操作来验证容器实例上的任务状态，以获取实例上的任务列表，然后使用带有每个任务的 Amazon Resource Name (ARN) 或 ID 的 DescribeTasks 操作来验证任务状态。

当您准备好容器实例再次开始托管任务时，您可以将容器实例的状态从 DRAINING 改为 ACTIVE。然后，Amazon ECS 服务调度程序将再次考虑容器实例放置任务。

过程

可以通过以下步骤使用新的 AWS Management Console 将容器实例设置为耗尽。

您还可以使用 UpdateContainerInstancesState API 操作或 update-container-instances-state 命令将容器实例的状态更改为 DRAINING。

AWS Management Console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面中，选择托管实例的集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Infrastructure（基础设施）选项卡。然后，在 Container instances（容器实例）下，选择要耗尽的每个容器实例的复选框。
5. 依次选择操作、耗尽。

Amazon ECS Linux 容器代理

Amazon ECS 代理是在向您的集群注册的每个容器实例上运行的进程。它可以促进您的容器实例与 Amazon ECS 之间的通信。

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。

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。

> **Important**
>
> 可靠指标的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更高版本，该版本包含在经 Amazon ECS 优化的 AMI `20220607` 及更高版本中。
> 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。

生命周期

当 Amazon ECS 容器代理将 Amazon EC2 实例注册到集群时，Amazon EC2 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）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 容器代理会在未注销原始容器实例 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，则可以安全地注销列出为 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。

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 耗尽 Amazon ECS 容器实例。

如果您注销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。

> **Important**
>
> 您可以手动耗尽实例，也可以构建 自动扩缩组生命周期钩子以将实例状态设置为 DRAINING。有关 Auto Scaling 生命周期钩子的更多信息，请参阅 Amazon EC2 Auto Scaling 生命周期钩子。

经 Amazon ECS 优化的 AMI

经 Amazon ECS 优化的 AMI 的 Linux 变体使用 Amazon Linux 2 AMI 作为其基础。可以通过查询 Systems Manager Parameter Store API 来检索每个变体的 Amazon Linux 2 源 AMI 名称。有关更

多信息，请参阅 检索经 Amazon ECS 优化的 Linux AMI 元数据。从最新的经 Amazon ECS 优化的 Amazon Linux 2 AMI 启动容器实例时，您将收到最新的容器代理版本。要启动具有经 Amazon ECS 优化的 Amazon Linux 2 AMI 的容器实例，请参阅 启动 Amazon ECS Linux 容器实例。

其他信息

以下页面提供了有关更改的更多信息：

- GitHub 上的 Amazon ECS 代理变更日志
- `ecs-init` 应用程序的源代码以及用于打包代理的脚本和配置现在已成为代理存储库的一部分。有关 `ecs-init` 的旧版本和包装，请参阅 GitHub 上的 Amazon ecs-init 变更日志
- Amazon Linux 2 发行说明。
- Docker 文档中的 Docker 引擎版本注释
- NVIDIA 文档中的 NVIDIA 驱动程序文档

Amazon ECS 容器代理配置

Amazon ECS 容器代理支持很多配置选项，其中大多数选项可通过环境变量来设置。

如果容器实例是使用经 Amazon ECS 优化的 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 引导启动 Amazon ECS Linux 容器实例以传递数据。

如果容器实例是使用经 Amazon ECS 优化的 AMI 的 Windows 变体启动的，则可以使用 PowerShell SetEnvironmentVariable 命令设置这些环境变量，然后重新启动代理。有关更多信息，请参阅《Amazon EC2 用户指南》中的启动时在 Windows 实例上运行命令和 the section called “引导启动容器实例”。

如果要手动启动 Amazon EC2 容器代理（对于没有经 Amazon ECS 优化的 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。将这些变量与语法 `--env=`*`VARIABLE_NAME`*`=`*`VARIABLE_VALUE`* 结合使用。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。您可以使用以下命令来添加变量。

```
sudo systemctl stop ecs
sudo vi /etc/ecs/ecs.config
```

```
# And add the environment variables with VARIABLE_NAME=VARIABLE_VALUE format.
sudo systemctl start ecs
```

可用参数

有关可用的 Amazon ECS 容器代理配置参数的信息，请参阅 GitHub 上的 Amazon ECS 容器代理。

将 Amazon ECS 容器实例配置存储在 Amazon S3 中

Amazon ECS 容器代理配置通过环境变量来控制。经 Amazon ECS 优化的 AMI 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能够在 `.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。可以将 `ecs.config` 文件的副本存储在私有存储桶中。然后，您可以使用 Amazon EC2 用户数据安装 AWS CLI，并在实例启动时将配置信息复制到 `/etc/ecs/ecs.config`。

要在 Amazon S3 中储存 **`ecs.config`** 文件

1. 您必须向容器实例角色（ecsInstanceRole）授予只读访问 Amazon S3 的权限。为此，您可以将 AmazonS3ReadOnlyAccess 分配给 `ecsInstanceRole` 角色。有关如何将策略附加到角色的信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色权限策略（控制台）
2. 使用以下格式创建包含有效 Amazon ECS 代理配置变量的 `ecs.config` 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 在 Amazon ECS 中使用非 AWS 容器映像。

   ```
   ECS_ENGINE_AUTH_TYPE=dockercfg
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
   ```

   > **Note**
   >
   > 有关可用 Amazon ECS 代理配置变量的完整列表，请参阅 GitHub 上的 Amazon ECS 容器代理。

3. 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 用户指南中的创建存储桶。
4. 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 用户指南中的向存储桶添加对象。

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. 启动新的容器实例，并在 EC2 用户数据中使用以下示例脚本。脚本将安装 AWS CLI 并将您的配置文件复制到 `/etc/ecs/ecs.config`。有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

安装 Amazon ECS 容器代理

如果您想向 Amazon ECS 集群注册 Amazon EC2 实例，并且该实例不使用基于经 Amazon ECS 优化的 AMI 的 AMI，您可以使用以下过程手动安装 Amazon ECS 容器代理。为此，您可以从区域性 Amazon S3 存储桶之一或 Amazon Elastic Container Registry Public 下载代理。如果您从区域性 Amazon 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。

> **Note**
>
> Amazon ECS 和 Docker 服务的 `systemd` 单元都有一个指令，在启动两个这两项服务之前等待 `cloud-init` 完成。在您的 Amazon EC2 用户数据完成运行之前，`cloud-init` 过程不会被视为已完成。因此，通过 Amazon EC2 用户数据启动 Amazon ECS 或 Docker 可能会导致死锁。要使用 Amazon EC2 用户数据启动容器代理，您可以使用 `systemctl enable --now --no-block ecs.service`。

在非 Amazon Linux EC2 实例上安装 Amazon ECS 容器代理

要在非 Amazon EC2 实例上安装 Amazon ECS 容器代理，您可以从区域性 Amazon S3 存储桶之一下载代理并安装它。

> **Note**
>
> 使用非 Amazon Linux AMI 时，您的 Amazon EC2 实例需要 `cgroupfs` 支持 `cgroup` 驱动程序，以便 Amazon ECS 代理能够支持任务级别的资源限制。有关更多信息，请参阅 GitHub 上的 Amazon ECS 代理。

下面按区域列出了每个系统架构最新的 Amazon ECS 容器代理文件，以供参考。

区域	区域名称	Amazon ECS init deb 文件	Amazon ECS init rpm 文件
us-east-2	美国东部（俄亥俄州）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
us-east-1	美国东部（弗吉尼亚州北部）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
us-west-1	美国西部（加利福尼亚北部）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
us-west-2	美国西部（俄勒冈州）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
ap-east-1	亚太地区（香港）	Amazon ECS init amd64（amd64）	Amazon ECS init x86_64（x86_64）

区域	区域名称	Amazon ECS init deb 文件	Amazon ECS init rpm 文件
		Amazon ECS init arm64（arm64）	Amazon ECS init aarch64（aarch64）
ap-northeast-1	亚太地区（东京)	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
ap-northeast-2	亚太地区（首尔）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
ap-south-1	亚太地区（孟买）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
ap-southeast-1	亚太地区（新加坡）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
ap-southeast-2	亚太地区（悉尼）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）

区域	区域名称	Amazon ECS init deb 文件	Amazon ECS init rpm 文件
ca-central-1	加拿大（中部）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
eu-central-1	欧洲地区（法兰克福）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
eu-west-1	欧洲地区（爱尔兰）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
eu-west-2	欧洲地区（伦敦）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
eu-west-3	欧洲地区（巴黎）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
sa-east-1	南美洲（圣保罗）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64 Amazon ECS init aarch64（aarch64）

区域	区域名称	Amazon ECS init deb 文件	Amazon ECS init rpm 文件
us-gov-east-1	AWS GovCloud（美国东部）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）
us-gov-west-1	AWS GovCloud（美国西部）	Amazon ECS init amd64（amd64） Amazon ECS init arm64（arm64）	Amazon ECS init x86_64（x86_64） Amazon ECS init aarch64（aarch64）

使用非 Amazon Linux AMI 在 Amazon 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。
2. 连接到您的实例。
3. 在实例上安装最新版本的 Docker。
4. 检查 Docker 版本以验证系统是否满足最低版本要求。

> **Note**
>
> 可靠指标的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更高版本，该版本包含在经 Amazon ECS 优化的 AMI `20220607` 及更高版本中。
> 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。

```
docker --version
```

5. 下载适用于您的操作系统和系统架构的相应的 Amazon ECS 代理文件并进行安装。

   对于 deb 架构：

```
ubuntu:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.amd64.deb
ubuntu:~$ sudo dpkg -i amazon-ecs-init-latest.amd64.deb
```

对于 rpm 架构：

```
fedora:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/
amazon-ecs-init-latest.x86_64.rpm
fedora:~$ sudo yum localinstall -y amazon-ecs-init-latest.x86_64.rpm
```

6. 编辑 /lib/systemd/system/ecs.service 文件并在 [Unit] 部分末尾添加以下行。

```
After=cloud-final.service
```

7. （可选）向 default 集群以外的集群注册实例，编辑 /etc/ecs/ecs.config 文件并添加以下内容。下面的示例指定了 MyCluster 集群。

```
ECS_CLUSTER=MyCluster
```

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。

> **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 Amazon ECS 容器实例配置存储在 Amazon S3 中 和在 Amazon ECS 中使用非 AWS 容器映像。

8. 启动 ecs 服务。

```
ubuntu:~$ sudo systemctl start ecs
```

使用主机网络模式运行 Amazon ECS 代理

在运行 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 实例元数据服务端点 (`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 Amazon ECS 任务 IAM 角色。

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 容器代理日志配置参数

Amazon ECS 容器代理将日志存储在您的容器实例上。

对于容器代理版本 1.36.0 及更高版本，预设情况下，日志位于 `/var/log/ecs/ecs-agent.log`（在 Linux 实例上）和 `C:\ProgramData\Amazon\ECS\log\ecs-agent.log`（在 Windows 实例上）。

对于容器代理版本 1.35.0 及以前的版本，预设情况下，日志位于 `/var/log/ecs/ecs-agent.log.`*`timestamp`*（在 Linux 实例上）和 `C:\ProgramData\Amazon\ECS\log\ecs-agent.log.`*`timestamp`*（在 Windows 实例上）。

预设情况下，代理日志每小时轮换一次，最多存储 24 个日志。

以下是可用于更改原定设置代理日志记录行为的容器代理配置变量。有关更多信息，请参阅 Amazon ECS 容器代理配置。

`ECS_LOGFILE`

示例值：`/ecs-agent.log`

Linux 上的原定设置值：Null

Windows 上的原定设置值：Null

代理日志应写入的位置。如果您通过 `ecs-init` 运行代理（这是使用经 Amazon ECS 优化的 AMI 时的默认方法），则容器内路径为 `/log`，并且 `ecs-init` 会将其挂载到主机上的 `/var/log/ecs/`。

`ECS_LOGLEVEL`

示例值：`crit`、`error`、`warn`、`info`、`debug`

Linux 上的原定设置值：`info`

Windows 上的原定设置值：`info`

要记录的详细级别。

ECS_LOGLEVEL_ON_INSTANCE

示例值：none、crit、error、warn、info、debug

Linux 上的原定设置值：none，如果 ECS_LOG_DRIVER 显式设置为非空值；否则，和 ECS_LOGLEVEL 相同

Windows 上的原定设置值：none，如果 ECS_LOG_DRIVER 显式设置为非空值；否则，与 ECS_LOGLEVEL 相同

可用于覆盖 ECS_LOGLEVEL 并设置应记录在实例日志文件中的详细级别，与日志记录驱动程序中记录的级别分开。如果显式设置了日志记录驱动程序，则默认情况下会关闭实例上的日志。可以使用此变量将其重新打开。

ECS_LOG_DRIVER

示例值：awslogs、fluentd、gelf、json-file、journald、logentries、syslog、splunk

Linux 上的原定设置值：json-file

Windows 上的默认值：不适用

确定代理容器使用的日志记录驱动程序。

ECS_LOG_ROLLOVER_TYPE

示例值：size、hourly

Linux 上的原定设置值：hourly

Windows 上的原定设置值：hourly

确定容器代理日志文件是按小时轮换还是基于大小轮换。预设情况下，代理日志文件每小时轮换一次。

ECS_LOG_OUTPUT_FORMAT

示例值：logfmt、json

Linux 上的原定设置值：logfmt

Windows 上的原定设置值：logfmt

确定日志输出格式。使用 json 格式时，日志中的每一行都是一个结构化的 JSON 映射。

ECS_LOG_MAX_FILE_SIZE_MB

示例值：10

Linux 上的原定设置值：10

Windows 上的原定设置值：10

将 ECS_LOG_ROLLOVER_TYPE 变量设置为 size 时，此变量确定日志文件在轮换之前的最大大小（以 MB 为单位）。如果将轮换类型设置为 hourly，则忽略此变量。

ECS_LOG_MAX_ROLL_COUNT

示例值：24

Linux 上的原定设置值：24

Windows 上的原定设置值：24

确定要保留的轮换日志文件的数量。达到此限制后，将删除较旧的日志文件。

对于容器代理版本 1.36.0 及更高版本，以下是使用 logfmt 格式时的示例日志文件。

```
level=info time=2019-12-12T23:43:29Z msg="Loading configuration" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-agent:latest" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-pause:0.1.0" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Amazon ECS agent Version: 1.36.0, Commit:
 ca640387" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Creating root ecs cgroup: /ecs"
 module=init_linux.go
level=info time=2019-12-12T23:43:29Z msg="Creating cgroup /ecs"
 module=cgroup_controller_linux.go
level=info time=2019-12-12T23:43:29Z msg="Loading state!" module=statemanager.go
level=info time=2019-12-12T23:43:29Z msg="Event stream ContainerChange start
 listening..." module=eventstream.go
level=info time=2019-12-12T23:43:29Z msg="Restored cluster 'auto-robc'" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Restored from checkpoint file. I
 am running as 'arn:aws:ecs:us-west-2:0123456789:container-instance/auto-
robc/3330a8a91d15464ea30662d5840164cd' in cluster 'auto-robc'" module=agent.go
```

以下是使用 JSON 格式时的示例日志文件。

```
{"time": "2019-11-07T22:52:02Z", "level": "info", "msg": "Starting Amazon Elastic
 Container Service Agent", "module": "engine.go"}
```

对于容器代理版本 1.35.0 及以前的版本，以下是日志文件的格式。

```
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"
```

为私有 Docker 映像配置 Amazon ECS 容器实例

Amazon ECS 容器代理可使用基本身份验证对私有注册表进行身份验证。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 在 Amazon ECS 中使用非 AWS 容器映像。

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。

经 Amazon ECS 优化的 AMI 的 Linux 变体在容器实例启动时以及每次服务启动时（使用 sudo start ecs 命令）扫描 `/etc/ecs/ecs.config` 文件以查找这些变量。没有经 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 Amazon ECS 容器实例配置存储在 Amazon S3 中。

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（对于经Amazon ECS 优化过的 AMI，该文件为 `/etc/ecs/ecs.config`）。如果不使用经 Amazon ECS 优化的 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（对于经 Amazon ECS 优化过的 AMI，该文件为 `/etc/ecs/ecs.config`）。如果不使用经 Amazon ECS 优化过的 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
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

过程

使用以下过程可为容器实例启用私有注册表。

在经 Amazon ECS 优化的 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。

4. 停止 ecs 服务：

```
sudo stop ecs
```

输出：

```
ecs stop/waiting
```

5. 重新启动 ecs 服务。

   - 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI：

     ```
     sudo systemctl restart ecs
     ```

   - 对于经 Amazon ECS 优化的 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 “容器自检”。

```
curl http://localhost:51678/v1/metadata
```

Amazon ECS 任务和映像自动清理

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。

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。

原定设置情况下，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。

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。

`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 秒钟。如果将该值设置为 1 秒以内，则系统会忽略该参数。

`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关闭自动映像清除，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。

当 Amazon ECS 容器代理正在运行且自动映像清除未关闭时，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。

在 Amazon ECS 上计划您的容器

Amazon Elastic Container Service（Amazon ECS）是一个共享状态的乐观并发系统，可为您的容器化工作负载提供灵活的计划功能。Amazon ECS 调度器利用由 Amazon ECS API 提供的相同群集状态信息来制定适当的放置决策。

Amazon ECS 提供用于长期运行的任务和应用程序的服务调度器。它还允许运行独立任务或计划任务（用于批处理作业或单个运行任务）。您可以为运行最能满足您需求的任务指定任务放置策略和约束。例如，您可以指定任务是跨多个可用区运行，还是在单个可用区内运行。您还可以选择将任务集成到自己的自定义或第三方调度器。

选项	何时使用	更多信息
服务	服务计划程序适用于长时间运行的无状态服务和应用程序。此外，服务调度器可以选择性地确保针对 Elastic Load Balancing 负载均衡器注册任务。您可以更新由服务调度器维护的服务。这可能包括部署新的任务定义或更改正在运行的所需任务的数量。预设情况下，服务调度器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。	Amazon ECS 服务
独立任务	独立任务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。使用 RunTask，您可以允许原定设置任务放置策略在集群中随机分配任务。这可以最大程度	Amazon ECS 独立任务

选项	何时使用	更多信息
	地减小单一实例获得不成比例数量任务的机会。	
计划任务	当您需要在集群中以设定的时间间隔运行任务时，计划任务是适合的，您可以使用 EventBridge Scheduler 创建计划。您可以为备份操作或日志扫描运行任务。您创建的 EventBridge Scheduler 计划可以在指定时间在集群中运行一个或多个任务。您的调度事件可以设置为特定的时间间隔（每 *N* 分钟、小时或天运行一次）。否则，对于更复杂的调度，您可以使用 `cron` 表达式。	使用 Amazon EventBridge 调度器计划 Amazon ECS 任务

计算选项

通过 Amazon ECS，您可以指定运行任务或服务的基础设施。您可以使用容量提供程序策略或启动类型。

对于 Fargate，容量提供程序是 Fargate 和 Fargate Spot。对于 EC2，容量提供程序是具有已注册容器实例的自动扩缩组。

容量提供程序策略将您的任务分配给与您的集群关联的容量提供程序。

只有已与集群关联且状态为 `ACTIVE` 或 `UPDATING` 的容量提供程序才能在容量提供程序策略中使用。在创建集群时，您可以将容量提供程序与集群相关联。

在容量提供程序策略中，可选基准值指明在指定的容量提供程序上至少运行多少个任务。在一个容量提供程序策略中，只能有一个容量提供程序策略定义了基准。

权重值指明使用指定容量提供程序的已启动任务总数的相对百分比。考虑以下示例。您的策略包含两个容量提供程序，并且两者的权重均为 1。当满足基准百分比时，任务会在两个容量提供程序之间均匀分

配。按照相同的逻辑，假定您指定 capacityProviderA 的权重为 1，并指定 capacityProviderB 的权重为 4，那么，对于使用 capacityProviderA 运行的每个任务，都会有四个任务使用 capacityProviderB。

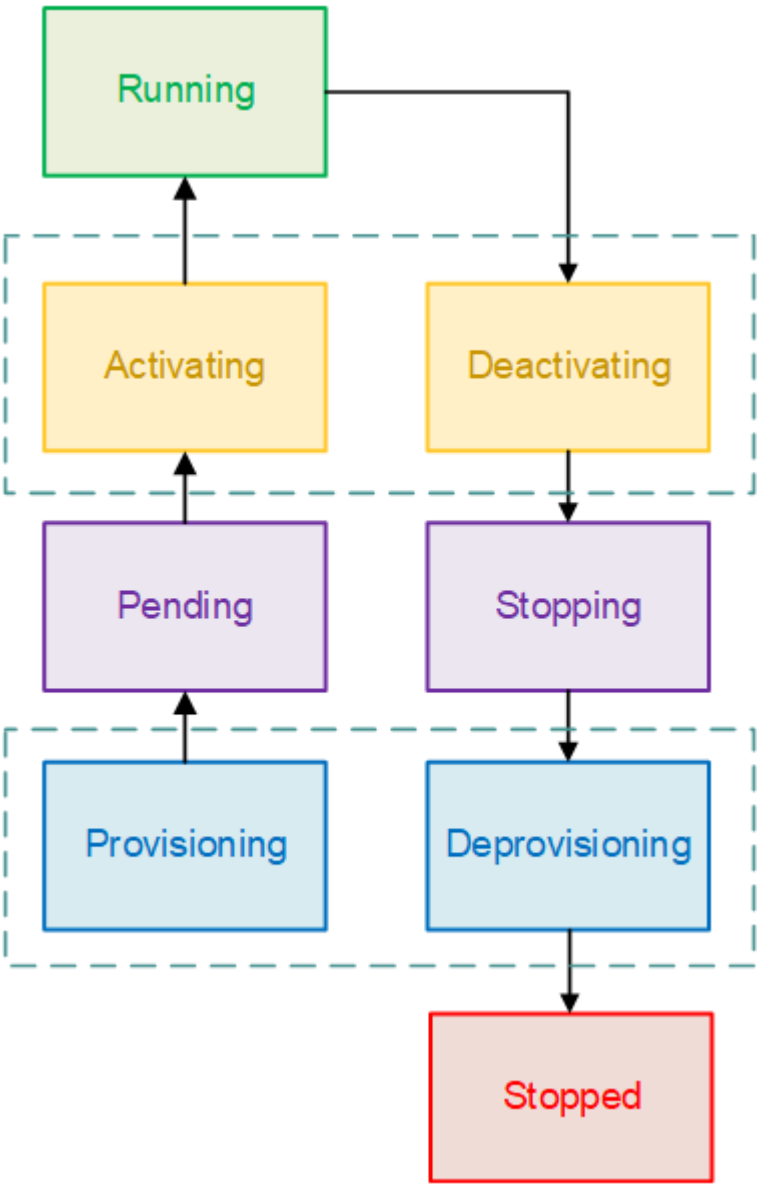
启动类型会直接在 Fargate 或您手动注册到集群的 Amazon EC2 实例上启动您的任务。

Amazon ECS 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

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。任务一直处于待处理状态，直到有可用资源可用于该任务。

ACTIVATING (正在激活)

这种一种过渡状态，在此状态下，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)

这种一种过渡状态，在此状态下，Amazon ECS 必须在任务停止前执行其他步骤。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。

对于 Linux 容器，容器代理将发送 SIGTERM 信号，以通知应用程序需要完成并关闭，然后在等待任务定义中设置的 StopTimeout 期限后发送 SIGKILL。

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。

如果您的任务因某个错误而停止，请参阅 查看 Amazon ECS 已停止任务错误。

DELETED

这是任务停止时的过渡状态。此状态不会显示在控制台中，而是显示在 describe-tasks 中。

Amazon ECS 如何将任务放置在容器实例上

您可以使用任务放置来配置 Amazon ECS，以将您的任务放在满足特定标准（例如可用区或实例类型）的容器实例上。

以下是任务放置组件：

- 任务放置策略是 - 用于选择放置任务的容器实例或要终止的任务的算法。例如，Amazon ECS 可以随机选择容器实例；还可以选择一组容器实例，将任务平均分配到其中。

- 任务组 - 一组相关的任务，例如数据库任务。
- 任务放置约束 - 必须满足这些规则才能在容器实例上放置任务。如果不满足该约束条件，则任务不会被放置并且会保持 PENDING 状态。例如，您可以使用约束条件将任务仅放置在特定的实例类型上。

Amazon ECS 对启动类型有不同的算法。

EC2 启动类型

对于使用 EC2 启动类型的任务，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。

原定设置任务放置策略取决于您是手动运行任务（独立任务）还是在服务中运行任务。对于作为 Amazon ECS 服务的一部分运行的任务，任务放置策略是使用 `attribute:ecs.availability-zone` 的 `spread`。服务中的任务没有默认的任务放置约束。有关更多信息，请参阅 在 Amazon ECS 上计划您的容器。

> **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，您可以使用任务放置策略和任务放置约束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. 识别满足任务定义中对 CPU、GPU、内存和端口要求的容器实例。
2. 识别满足任务放置约束的容器实例。
3. 识别满足任务放置策略的容器实例。
4. 选择放置任务的容器实例。

Fargate 启动类型

使用 Fargate 启动类型的任务不支持任务放置策略和约束。Fargate 将尽最大努力将任务分散到各个可访问的可用区。如果容量提供程序同时包含 Fargate 和 Fargate Spot，则每个容量提供程序的分散行为是独立的。

使用策略来定义 Amazon ECS 任务放置

对于使用 EC2 启动类型的任务，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。

原定设置任务放置策略取决于您是手动运行任务（独立任务）还是在服务中运行任务。对于作为 Amazon ECS 服务的一部分运行的任务，任务放置策略是使用 `attribute:ecs.availability-zone` 的 `spread`。服务中的任务没有默认的任务放置约束。有关更多信息，请参阅 在 Amazon ECS 上计划您的容器。

> **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，您可以使用任务放置策略和任务放置约束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. 识别满足任务定义中对 CPU、GPU、内存和端口要求的容器实例。
2. 识别满足任务放置约束的容器实例。
3. 识别满足任务放置策略的容器实例。
4. 选择放置任务的容器实例。

使用 `placementStrategy` 参数在服务定义或任务定义中指定任务放置策略。

```
"placementStrategy": [
    {
        "field": "The field to apply the placement strategy against",
        "type": "The placement strategy to use"
    }
]
```

您可以在运行任务（RunTask）、创建新服务（CreateService）或更新现有服务（UpdateService）时指定策略。

下表介绍了可用的类型和字段。

type	有效的字段值	
`binpack` 将任务放置在容器实例上，以保留最少的未使用 CPU 或内存。此策略最大限度地减少了正在使用的容器实例的数量。 当使用此策略并执行横向缩减操作时，Amazon ECS 将终止任务。它根据任务终止后留在容器实例上的资源量执行此操作。任务终止后剩下可用资源最多的容器实例将终止该任务。	• cpu • memory	
`random` 任务随机放置。	未使用	
`spread` 根据指定的值均匀放置任务。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。有关任务组的更多信息，请参阅 与组相关的 Amazon ECS 任务 。 当使用 `spread` 策略并采取横向缩减行动时，Amazon ECS 将选择要终止的任务，以保持可用性区域之间的平衡。在可用区域内，将随机选择任务。	• `instanceId` （或 `host`，效果相同） • 应用于容器实例的任何平台或自定义属性，例如 `attribute:ecs.availability-zone`	

也可以针对现有服务更新任务放置策略。有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

您可以按照您希望的执行顺序创建策略阵列，从而创建使用多种策略的任务置放策略。例如，如果您想将任务分散到多个可用区，然后根据每个可用区内的内存装填任务，请指定可用区策略，然后指定内存策略。有关策略示例，请参阅 Amazon ECS 任务放置策略示例。

Amazon ECS 任务放置策略示例

您可以使用以下操作指定任务放置策略：CreateService、UpdateService 和 RunTask。

示例

- 跨可用区平均分配任务
- 在所有实例上平均分配任务
- 基于内存的垃圾箱任务
- 随机置放任务
- 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
- 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
- 在实例之间平均分配任务，然后根据内存装填任务

跨可用区平均分配任务

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"
    }
]
```

在所有实例上平均分配任务

以下策略可在所有实例中平均分配任务。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
```

```
    }
]
```

基于内存的垃圾箱任务

以下策略根据内存装填任务。

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

随机置放任务

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

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"
    }
]
```

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

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"
    }
]
```

在实例之间平均分配任务，然后根据内存装填任务

以下策略将任务平均分配给所有实例，然后根据每个实例中的内存装填任务。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

与组相关的 Amazon ECS 任务

您可以识别一组相关任务并将其放置在任务组中。使用 `spread` 任务放置策略时，将具有相同任务组名称的所有任务视为一个集合。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。要确保数据库在可用性区域之间保持平衡，请将它们添加到名为 `databases` 的任务组中，然后使用 `spread` 任务放置策略。有关更多信息，请参阅 使用策略来定义 Amazon ECS 任务放置。

任务组也可以用作任务放置约束。在 `memberOf` 约束中指定任务组时，任务仅发送到在指定任务组中运行任务的容器实例。有关示例，请参阅Amazon ECS 任务放置约束示例。

预设情况下，独立任务使用任务定义系列名称（例如 `family:my-task-definition`）作为任务组名称（如果未指定自定义任务组名称）。作为服务的一部分启动的任务使用服务名称作为任务组名称，不能更改。

以下任务组要求适用。

- 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。

定义 Amazon ECS 将哪些容器实例用于任务

任务放置约束是关于容器实例的规则，Amazon ECS 使用该规则来确定是否允许在实例上运行任务。必须至少有一个容器实例匹配约束条件。如果没有与约束条件匹配的实例，则任务将保持 PENDING 状态。创建新服务或更新现有服务时，您可以为服务的任务指定任务放置限制。

您可以使用 `placementConstraint` 参数在服务定义、任务定义或任务中指定任务放置约束。

```
"placementConstraint": [
    {
        "expression": "The expression that defines the task placement constraints",
        "type": "The placement constraint type to use"
    }
]
```

下表介绍如何使用这些参数。

Constraint type	可以在何时指定
`distinctInstance` 将每项任务放置在不同的容器实例中。 **Important** 建议为其任务寻求强隔离的客户使用 Fargate。Fargate 在硬件虚拟化环境中运行每个任务。这将确保这些容器化工作负载不会与其他任务共享网络接口、Fargate 临时存	• 运行任务 RunTask • 创建新服务 CreateService，

Constraint type	可以在何时指定	
储、CPU 或内存。有关更多信息，请参阅 AWS Fargate 的安全概述。		
memberOf 将任务放置在满足表达式的容器实例中。	• 运行任务 RunTask • 创建新服务 CreateService， • 创建新的任务定义 RegisterTaskDefinition • 创建任务定义 RegisterTaskDefinition 的新修订 • 更新服务 UpdateService	

使用 memberOf 约束类型时，您可以使用集群查询语言创建表达式，该语言定义了 Amazon ECS 在其中放置任务的容器实例。表达式是供您按属性对容器实例进行分组的一种方式。表达式位于 placementConstraint 的 expression 参数中。

Amazon ECS 容器实例属性

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。

以下部分包含示例内置、可选属性和自定义属性。

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-1234abcd。

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。此属性的可能值为 `linux` 和 `windows`。

`ecs.os-family`

实例的操作系统版本。

对于 Linux 实例，有效值为 `LINUX`。对于 Windows 实例，ECS 以 `WINDOWS_SERVER_<OS_Release>_<FULL or CORE>` 格式设置值。有效值为 `WINDOWS_SERVER_2022_FULL`、`WINDOWS_SERVER_2022_CORE`、`WINDOWS_SERVER_20H2_CORE`、`W` 和 `WINDOWS_SERVER_2016_FULL`。

这对于 Windows 容器和 Windows containers on AWS Fargate 很重要，因为每个 Windows 容器的操作系统版本都必须与主机的操作系统版本相匹配。如果容器映像的 Windows 版本与主机不同，则容器无法启动。有关更多信息，请参阅 Microsoft 文档网站上的 Windows 容器版本兼容性。

如果您的集群运行多个 Windows 版本，则可以使用置放约束来确保将任务放在运行相同版本的 EC2 实例上：`memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)`。有关更多信息，请参阅 the section called “检索经 Amazon ECS 优化的 Windows AMI 元数据”。

`ecs.cpu-architecture`

实例的 CPU 架构。本属性的示例值为 `x86_64` 和 `arm64`。

`ecs.vpc-id`

启动实例的 VPC。本属性的示例值为 `vpc-1234abcd`。

`ecs.subnet-id`

实例正在使用的子网。本属性的示例值为 `subnet-1234abcd`。

可选属性

Amazon ECS 可能会将以下属性添加到您的容器实例。

`ecs.awsvpc-trunk-id`

如果此属性存在，则实例具有中继网络接口。有关更多信息，请参阅 增加 Amazon ECS Linux 容器实例网络接口。

`ecs.outpost-arn`

如果此属性存在，则包含 Outpost 的 Amazon Resource Name (ARN)。有关更多信息，请参阅 the section called “AWS Outposts 上的 Amazon Elastic Container Service”。

`ecs.capability.external`

如果此属性存在，则实例将标识为外部实例。有关更多信息，请参阅 外部启动类型的 Amazon ECS 集群。

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。

指定自定义属性时，必须考虑以下内容。

- `name` 必须包含 1 到 128 个字符，名称可以包含字母（大写和小写形式）、数字、连字符、下划线、正斜杠、反斜杠或句点。
- `value` 必须包含 1 到 128 个字符，可以包含字母（大写和小写形式）、数字、连字符、下划线、句点、符号 (@)、正斜杠、反斜杠、冒号或空格。该值不能包含任何前导空格或尾随空格。

创建表达式，以为 Amazon ECS 任务定义容器实例

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 Amazon ECS 容器实例属性。

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 将应用程序作为 Amazon ECS 任务运行 和 使用控制台创建 Amazon ECS 服务。您还可以在列出容器实例时应用组筛选条件。

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 Amazon ECS 容器实例属性。

`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 与组相关的 Amazon ECS 任务 。

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	Greater than
>=, greater_than_equal	大于或等于
<, less_than	Less than
<=, less_than_equal	小于或等于
exists	主题存在
!exists, not_exists	主题不存在
in	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数据库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']
```

Amazon ECS 任务放置约束示例

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、UpdateService、RegisterTaskDefinition 和 RunTask。

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

该示例使用 memberOf 约束将任务放置在进程守护程序服务 daemon-service 任务组中具有任务的实例上，同时考虑到同时指定的任何任务放置策略。此约束可确保进程守护程序服务任务在副本服务任务之前放置在 EC2 实例上。

将 daemon-service 替换为进程守护程序服务的名称。

```
"placementConstraints": [
    {
        "expression": "task:group == service:daemon-service",
        "type": "memberOf"
    }
]
```

该示例使用 memberOf 约束将任务放置在 databases 任务组中具有其他任务的实例上，同时考虑到也指定的任何任务放置策略。有关任务组的更多信息，请参阅 与组相关的 Amazon ECS 任务 。可以使用以下操作指定此约束：CreateService、UpdateService、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、UpdateService 和 RunTask。

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
```

```
]
```

Amazon ECS 独立任务

如果您具有执行某些工作然后再停止（例如批处理进程）的应用程序，则您可以将应用程序作为任务运行。如果您想运行一次任务，则可以使用控制台、AWS CLI、API 或 SDK。

如果您需要按基于费率、基于 cron 的计划或一次性计划运行应用程序，则可以使用 EventBridge 计划程序创建计划。

任务工作流程

当您启动 Amazon ECS 任务（独立任务或 Amazon ECS 服务执行的任务）时，会创建一项任务且最初会将其移至 PROVISIONING 状态。当任务处于 PROVISIONING 状态时，任务和容器都不存在，因为 Amazon ECS 需要找到用于放置任务的计算容量。

Amazon ECS 会根据您的启动类型或容量提供程序配置为您的任务选择合适的计算容量。您可以在 Fargate 和 Amazon EC2 启动类型中使用容量提供程序和容量提供程序策略。使用 Fargate，您不必考虑集群容量的预调配、配置和扩展。Fargate 负责您的任务的所有基础设施管理。对于 EC2 启动类型，您可以通过将 Amazon EC2 实例注册到集群来管理集群容量，也可以使用集群自动扩缩来简化计算容量管理。集群自动扩缩负责动态扩展集群容量，以便您可以专注于正在运行的任务。Amazon ECS 根据您在任务定义中指定的要求（例如 CPU 和内存）以及您的放置约束和策略确定将任务放置在何处。有关更多信息，请参阅Amazon ECS 如何将任务放置在容器实例上。

如果您使用启用了托管式扩缩的容量提供程序，则由于缺乏计算容量而无法启动的任务将移至 PROVISIONING 状态，而不是立即失败。找到放置任务的容量后，Amazon ECS 会预调配必要的附件（例如，awsvpc 模式下任务的弹性网络接口（ENI））。它使用 Amazon ECS 容器代理提取您的容器映像，然后再启动您的容器。预调配完成且相关容器启动后，Amazon ECS 会将任务移至 RUNNING 状态。有关任务状态的信息，请参阅 Amazon ECS 任务生命周期。

优化 Amazon ECS 任务启动时间

要加快任务启动速度，请考虑以下建议。

- 缓存容器映像和装填实例

  如果您使用 EC2 启动类型，则可以将 Amazon ECS 容器代理的拉取行为配置为 ECS_IMAGE_PULL_BEHAVIOR：prefer-cached。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。为容器关闭自动映像清除，以确保不会删除缓存的映像。这样可以缩短后续

启动的映像拉取时间。当容器实例中的任务密度很高时，缓存的效果会更大，您可以使用 binpack 放置策略对其进行配置。缓存容器映像对于基于 Windows 的工作负载特别有益，这些工作负载通常有很大（数十 GB）的容器映像大小。使用 binpack 放置策略时，您还可以考虑使用弹性网络接口（ENI）中继在每个容器实例上使用 awsvpc 网络模式放置更多任务。ENI 中继增加了您可以在 awsvpc 模式下运行的任务数量。例如，可能仅支持同时运行 2 个任务的 c5.large 实例通过 ENI 中继最多可运行 10 个任务。

- 选择最佳网络模式

  尽管在许多情况下，awsvpc 网络模式是理想的，但这种网络模式本质上会增加任务启动延迟，因为对于 awsvpc 模式下的每个任务，Amazon ECS 工作流都需要通过调用 Amazon EC2 API 来预预调配和附加 ENI，这会给您的任务启动增加几秒钟的开销。相比之下，使用 awsvpc 网络模式的主要优势是每个任务都有一个允许或拒绝流量的安全组。这意味着，您可以更灵活地在更精细的级别上控制任务与服务之间的通信。如果您优先考虑部署速度，则可以考虑使用 bridge 模式来加快任务启动速度。有关更多信息，请参阅 the section called “AWSVPC 网络模式”。

- 跟踪您的任务启动生命周期，以寻找优化机会

  通常很难知道应用程序启动所需的时间量。在应用程序启动期间启动容器映像、运行启动脚本和其他配置可能需要大量时间。您可以使用任务元数据端点发布指标，以跟踪从 ContainerStartTime 到应用程序准备好提供流量的应用程序启动时间。利用这些数据，您可以了解您的应用程序对总启动时间的贡献度，并找到可以减少不必要的应用程序特定开销和优化容器映像的领域。有关更多信息，请参阅 优化 Amazon ECS 容量和可用性。

- 选择最佳实例类型（对于 EC2 启动类型）

  选择正确的实例类型基于您在任务中配置的资源预留（例如 CPU、内存）。因此，在调整实例大小时，您可以计算在单个实例上可以放置多少任务。放置良好的任务的一个简单例子是，在 m5.large 实例（支持 2 vCPU 和 8GB 内存）中托管 4 个任务，需要 0.5 vCPU 和 2GB 内存预留。此任务定义的预留充分利用了实例的资源。

将应用程序作为 Amazon ECS 任务运行

您可以使用 AWS Management Console 为一次性过程创建任务。

要创建独立任务（AWS Management Console）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. Amazon ECS 控制台允许您通过集群详细信息页面或任务定义修订列表创建独立任务。根据您选择的资源页面，使用以下步骤来创建您的独立任务。

从中启动服务	步骤
集群详细信息页面...	a. 在集群页面上，选择要在其中创建服务的集群。 b. 从任务选项卡上，选择运行新任务。
任务定义修订页面...	a. 在任务定义页面上，选择任务定义系列以显示该系列的修订。 b. 选择要使用的修订。 c. 从部署菜单中选择运行任务。

3. （可选）计算配置（高级）部分供您选择任务的分配方式。您可以使用容量提供程序策略或启动类型。要使用容量提供程序策略，您必须在集群级别配置容量提供程序。如果您尚未将集群配置为使用容量提供程序，则请改用启动类型。

分配方式	步骤
容量提供程序策略	a. 在 Compute options（计算选项）部分中，选择 Capacity provider strategy（容量提供程序策略）。 b. 选择策略： • 要使用集群的原定设置容量提供程序策略，请选择 Use cluster default（使用集群原定设置）。 • 如果集群没有原定设置的容量提供程序策略，或者要使用自定义策略，请选择 Use

分配方式	步骤
	custom（使用自定义），Add capacity provider strategy（添加容量提供程序策略），并通过指定 Base（基准）、Capacity provider（容量提供程序）和 Weight（权重）来定义自定义容量提供程序策略。 **Note** 要在策略中使用容量提供程序，容量提供程序必须与集群相关联。
启动类型	a. 在 Compute options（计算选项）部分中，选择 Launch type（启动类型）。 b. 对于 Launch type（启动类型），请选择一种启动类型。 c. （可选）当指定 Fargate 启动类型时，对于平台版本中，指定要使用的平台版本。如果未指定任何版本，将使用 LATEST 平台版本。

4. 对于应用程序类型，选择任务。
5. 对于任务定义，选择任务定义系列和修订。

> **Important**
>
> 控制台验证选择内容，确保所选任务定义系列和修订版与定义的计算配置兼容。

6. 对于 Desired tasks（预期任务），请输入要启动的任务数量。
7. 如果您的任务定义使用 awsvpc 网络模式，请展开 Networking（联网）。使用以下步骤指定自定义配置。

   a. 对于 VPC，选择要使用的 VPC。

   b. 对于 Subnets（子网），选择 VPC 中的一个或多个子网，任务计划程序在放置任务时会考虑这些子网。

   > **Important**
   >
   > awsvpc 网络模式仅支持私有子网。任务不接收公有 IP 地址。因此，出站互联网访问需要 NAT 网关，且入站互联网流量通过负载均衡器进行路由。

   c. 对于安全组，您可以选择现有安全组或创建新安全组。要使用现有安全组，请选择该安全组并移至下一步。要创建新安全组，请选择 Create a new security group (创建新安全组)。您必须指定安全组名称、说明，然后为该安全组添加一个或多个入站规则。

   d. 对于公有 IP，选择是否向任务的 elastic network interface (ENI) 自动分配公有 IP 地址。

   在公有子网中运行时，可以为 AWS Fargate 任务分配一个公有 IP 地址，以便它们具有通往互联网的路由。有关更多信息，请参阅 AWS Fargate Amazon Elastic Container Service 开发人员指南 中的 Fargate任务联网。
8. 如果您的任务使用的数据卷与部署时的配置兼容，则可以通过扩展卷来配置该卷。

卷名称和卷类型在创建任务定义修订时配置，在运行独立任务时无法更改。要更新卷名称和类型，您必须创建新的任务定义修订并使用新修订运行任务。

要配置此卷类型	请执行该操作
Amazon EBS	a. 为 EBS 卷类型选择要附加到任务的 EBS 卷的类型。 b. 为大小（GiB）输入卷大小的有效值，以吉字节（GiB

要配置此卷类型	请执行该操作	
	）为单位。您可以指定最小 1GiB 和最大 16384GiB 的卷大小。除非您提供快照 ID，否则此值为必填值。 c. 对于 IOPS，输入该卷应提供的输入/输出操作的次数（IOPS）的最大值。此值只能针对 io1、io2 和 gp3 卷类型进行配置。 d. 对于吞吐量（MiB/s），以每秒兆字节（MiBps 或 MiB/s）为单位输入卷应提供的吞吐量。此值只能针对 gp3 卷类型进行配置。 e. 对于快照 ID，请选择现有的 Amazon EBS 卷快照，或者如果您想从快照中创建卷，请输入快照的 ARN。您还可以通过不选择或输入快照 ID 来创建新的空卷。 f. 对于终止策略，如果您希望在任务终止后保留为附加到任务而配置的卷，则请取消选中该复选框。默认情况下，任务终止时，将会删除附加到任务的 EBS 卷。 g. 对于文件系统类型，请选择将用于在卷上数据存储和检索的文件系统的类型。您可以选择操作系统默认类型或特定的文件系	

要配置此卷类型	请执行该操作	
	统类型。Linux 的默认设置为 XFS。对于从快照创建的卷，必须指定创建快照时卷使用的相同文件系统类型。如果文件系统类型不匹配，则任务将无法启动。 h. 对于基础设施角色，请选择一个具有必要权限的 IAM 角色，以允许 Amazon ECS 管理任务的 Amazon EBS 卷。您可以将 `AmazonECSInfrastructureRolePolicyForVolumes` 托管策略附加到该角色，也可以使用策略作为指南，以创建并附加您自己的具有满足您特定需求的权限的策略。有关必要权限的更多信息，请参阅 Amazon ECS 基础设施 IAM 角色。 i. 对于加密，如果您想使用 Amazon EBS 默认加密设置，则请选择默认。如果您的账户配置了默认加密，则该卷将使用设置中指定的 AWS Key Management Service（AWS KMS）密钥进行加密。如果您选择默认，但未开启 Amazon EBS 默认	

要配置此卷类型	请执行该操作
	加密，则该卷将处于未加密状态。 如果选择自定义，则可以为卷加密指定您选择的 AWS KMS key。 如果选择无，则除非您配置了加密，或者您使用加密的快照创建了卷，否则该卷将处于未加密状态。 j. 如果您选择了自定义作为加密，则必须指定要使用的 AWS KMS key。为 KMS 密钥选择 AWS KMS key 或输入密钥 ARN。如果您选择使用对称的客户托管密钥对卷进行加密，请确保您拥有 AWS KMS key 策略中定义的正确权限。有关更多信息，请参阅 Amazon EBS 卷的数据加密。 k. （可选）在标签下，您可以通过传播任务定义中的标签或提供自己的标签来将标签添加到 Amazon EBS 卷中。 如果您要传播任务定义中的标签，请为从中传播标签选择任务定义。如果您选择不传播，或者如果您未选择值，则不会传播标签。

要配置此卷类型	请执行该操作	
	如果您想提供自己的标签，请选择添加标签，然后为您添加的每个标签提供密钥和值。 有关标记 Amazon EBS 卷的更多信息，请参阅标记 Amazon EBS 卷。	

9. （可选）要使用默认策略之外的其他任务放置策略，请展开 Task Placement（任务放置），然后从以下选项中进行选择。

   有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

   - 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 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。
   - 装填 – 根据 CPU 或内存的最低可用量来分配任务。
   - 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。
   - 自定义 – 定义您自己的任务放置策略。

   如果您选择了 Custom（自定义），请定义放置任务的算法和任务放置过程中要考虑的规则。

   - 在 Strategy（策略）下，对于 Type（类型）和 Field（字段），选择算法和用于该算法的实体。

     您最多可输入 5 个策略。
   - 在约束下，对于类型和表达式，选择要用于该约束的规则和属性。

     例如，要设置在 T2 实例上放置任务的约束，对于 Expression（表达式），输入 attribute:ecs.instance-type =~ t2.*。

     您最多可输入 10 个约束。

10. （可选）要覆盖任务定义中定义的任务 IAM 角色或任务执行角色，请展开 Task overrides（任务覆盖），然后完成以下步骤：

    a. 对于任务角色，请选择此任务的 IAM 角色。有关更多信息，请参阅 Amazon ECS 任务 IAM 角色。

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 角色的说明，请参阅 创建任务 IAM 角色。

b. 对于任务执行角色，请选择任务执行角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

11. （可选）要覆盖容器命令和环境变量，请展开 Container Overrides（容器覆盖），然后展开容器。

- 要向容器发送任务定义命令以外的命令，请在命令覆盖中输入 Docker 命令。

  有关 Docker 运行命令的更多信息，请参阅“Docker Reference Manual”（《Docker 参考手册》）中的 Docker Run reference（Docker 运行参考）。
- 要添加环境变量，请选择 Add Environment Variable（添加环境变量）。对于 Key，键入您的环境变量名称。对于值，输入环境值的字符串值（不使用双引号 (" ")）。

  AWS 用双引号（""）括起字符串，并按以下格式将字符串传递给容器：

```
MY_ENV_VAR="This variable contains a string."
```

12. （可选）为了帮助识别您的任务，请展开 Tags（标签）部分，然后配置您的标签。

要让 Amazon ECS 使用集群名称和任务定义标签自动标记全部新启动的任务，选择 Turn on Amazon ECS managed tags（启用 Amazon ECS 托管标签），然后选择 Task definitions（任务定义）。

添加或删除标签。

- [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
  - 对于 Key（键），输入键名称。
  - 对于值，输入键值。
- [删除标签] 在标签旁，选择 Remove tag (删除标签)。

13. 选择创建。

使用 Amazon EventBridge 调度器计划 Amazon ECS 任务

EventBridge 调度器是一个无服务器调度器，使您能够从一个中央托管服务创建、运行和管理任务。它提供独立于事件总线和规则的一次性和重复性计划功能。EventBridge Scheduler 具有高度可定制性，

与 EventBridge 计划规则相比，可扩展性更高，目标 API 操作和 AWS 服务范围更广。EventBridge Scheduler 提供以下计划，您可以在 EventBridge Scheduler 控制台中为您的任务配置这些计划：

- 基于速率
- 基于 Cron

  您可以在任何时区配置基于 cron 的计划。
- 一次性计划

  您可以在任何时区配置一次性的计划。

您可以使用 Amazon EventBridge 调度器计划 Amazon ECS。

尽管您可以在 Amazon ECS 控制台中创建计划任务，但目前 EventBridge 调度器控制台提供了更多功能。

在计划任务之前，完成以下步骤：

1. 使用 VPC 控制台，获取任务运行所在的子网 ID 和子网的安全组 ID。有关更多信息，请参阅《Amazon VPC 用户指南》中的查看您的子网和查看您的安全组。
2. 配置 EventBridge Scheduler 执行角色。有关更多信息，请参阅《Amazon EventBridge 调度器用户指南》中的设置执行角色。

要使用控制台创建新计划

1. 打开 Amazon EventBridge 调度器控制台，网址为：https://console.aws.amazon.com/scheduler/home。
2. 在计划页面，选择创建计划。
3. 在指定计划详细信息页面，在计划名称和描述部分中，执行以下操作：

   a. 对于计划名称，输入计划的名称。例如，**`MyTestSchedule`**。

   b. （可选）对于描述，输入对计划的描述。例如，**`TestSchedule`**。

   c. 对于计划组，请选择一个计划组。如果您没有计划组，选择默认。要创建计划组，选择创建自己的计划。

      您可以使用计划组将标签添加到计划组。
4. 选择计划选项。

出现	请执行此操作...	
一次性计划 一次性计划仅在您指定的日期和时间调用一次目标。	对于日期和时间，请执行以下操作： • 输入 YYYY/MM/DD 格式的有效日期。 • 输入 24 小时 hh:mm 格式的时间戳。 • 对于时区，选择时区。	

出现	请执行此操作...
定期计划 定期计划按照您使用 cron 表达式或 rate 表达式指定的速率调用目标。	a. 对于计划类型，执行以下操作之一： • 要使用 cron 表达式定义计划，请选择基于 cron 的计划并输入 cron 表达式。 • 要使用 rate 表达式定义计划，请选择基于 rate 的计划并输入 rate 表达式。 有关 cron 和 rate 表达式的更多信息，请参阅 Amazon EventBridge 调度器用户指南 中的 EventBridge 调度器的计划类型。 b. 对于灵活的时间窗口，选择关闭以关闭该选项，或者选择一个预定义的时间窗口。例如，如果您选择 15 分钟并且将定期计划设置为每小时调用一次其目标，则该计划将在每小时开始后的 15 分钟内运行。

5. （可选）如果您在上一步中选择定期计划，在时间范围部分，请执行以下操作：

   a. 对于时区，请选择时区。

   b. 对于开始日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

   c. 对于结束日期和时间，请输入 YYYY/MM/DD 格式的有效日期，然后指定 24 小时 hh:mm 格式的时间戳。

6. 选择下一步。
7. 在选择目标页面上，执行以下操作：
   a. 选择所有 API，然后在搜索框中输入 ECS。
   b. 选择 Amazon ECS。
   c. 在搜索框中输入 RunTask，然后选择 RunTask。
   d. 对于 Cluster，选择集群。
   e. 对于 ECS 任务，请选择要用于任务的任务定义。
   f. 要使用启动类型，请展开计算选项，然后选择启动类型。然后，选择启动类型。

      当指定 Fargate 启动类型时，对于平台版本中，输入要使用的平台版本。如果未指定任何平台，将使用 LATEST 平台版本。
   g. 对于子网，请输入要在其中运行任务的子网 ID。
   h. 对于安全组，请输入子网的安全组 ID。
   i. （可选）要使用默认策略之外的其他任务放置策略，请展开置放约束，然后输入约束。

      有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。
   j. （可选）为了帮助识别您的任务，请在标签下配置您的标签。

      要让 Amazon ECS 使用任务定义标签自动标记全部新启动的任务，选择启用 Amazon ECS 托管标签。
8. 选择下一步。
9. 在 Settings (设置) 页面上，执行以下操作：
   a. 要打开计划，在计划状态下，切换启用计划。
   b. 要为计划配置重试策略，在重试策略和死信队列（DLQ）下，请执行以下操作：
      - 切换重试。
      - 对于事件的最长保留时间，输入 EventBridge Scheduler 器必须保留未处理事件的最长小时和分钟数。
      - 最长时间为 24 小时。
      - 对于最大重试次数，输入在目标返回错误的情况下，EventBridge 调度器重试计划的最大次数。

        最大值为 185 次重试。

配置重试策略后，如果计划未能调用其目标，EventBridge 调度器将重新运行该计划。如果已配置，则必须为计划设置最长保留时间和最大重试次数。

c. 选择 EventBridge 调度器存储未送达事件的位置。

死信队列（DLQ）选项	请执行此操作...
请勿存储	选择 None。
将事件存储在创建计划所在的同一 AWS 账户 中	a. 选择在我的 AWS 账户 中选择一个 Amazon SQS 队列作为 DLQ。 b. 选择 Amazon SQS 队列的 Amazon 资源名称（ARN）。
将事件存储在与创建计划所在不同的 AWS 账户 中	a. 选择在另一个 AWS 账户 中指定一个 Amazon SQS 队列作为 DLQ。 b. 输入 Amazon SQS 队列的 Amazon 资源名称（ARN）。

d. 要使用客户托管密钥加密目标输入，在加密下，选择自定义加密设置（高级）。

如果选择此选项，请输入现有的 KMS 密钥 ARN 或选择创建一个 AWS KMS key 以导航到 AWS KMS 控制台。有关 EventBridge 调度器如何加密静态数据的更多信息，请参阅 Amazon EventBridge 调度器用户指南 中的 静态加密。

e. 对于权限，选择使用现有角色，然后选择角色。

要让 EventBridge 调度器为您创建新的执行角色，请选择为此计划创建新角色。然后，在角色名称中输入名称。如果您选择此选项，EventBridge 调度器会将模板化目标所需的必要权限附加到该角色。

10. 选择下一步。
11. 在查看并创建计划页面上，查看计划的详细信息。在每个部分中，选择编辑返回到该步骤并编辑其详细信息。

12. 选择创建计划。

    您可以在计划页面上查看新的和现有的计划列表。在状态列下，验证新计划是否已启用。

后续步骤

您可以使用 EventBridge Scheduler 控制台或 AWS CLI 来管理计划。有关更多信息，请参阅《Amazon EventBridge 调度器用户指南》中的管理计划。

停止 Amazon ECS 任务

如果您不再需要保持独立任务的运行状态，则可以停止该任务。Amazon ECS 控制台可以轻松停止一项或多项任务。

如果您想停止服务，请参阅使用控制台删除 Amazon ECS 服务。

要停止独立任务（AWS Management Console）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在集群页面上，选择要导航到集群详细信息页面的集群。
4. 在集群详细信息页面上，选择任务选项卡。
5. 您可以使用筛选启动类型列表按启动类型筛选任务。

要停止的任务	步骤
一个或多个	a. 选择任务，然后选择停止、停止选中任务。 b. 在停止任务确认页面上，选择停止
全部	**Important** 如果您选择使用控制台停止所有任务，则 Amazon ECS 会停止所有独立任务和属于服务一部分的任务。

要停止的任务	步骤
	因此，建议谨慎使用此选项。 a. 依次选择停止、全部停止。 b. 在停止任务确认页面上，输入停止所有任务，然后选择停止。

Amazon ECS 服务

您可以使用 Amazon ECS 服务在 Amazon ECS 集群中同时运行和维护指定数量的任务定义实例。如果其中一个任务失败或停止，Amazon ECS 服务调度器会启动另一个任务定义实例来替换它。这有助于保持服务中的预期任务数。

您也可选择在负载均衡器后面运行服务。负载均衡器将在与服务关联的各个任务间分配流量。

我们建议您将服务调度器用于长时间运行的无状态服务和应用程序。服务调度器可确保遵循您指定的计划策略并在任务失败时（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。例如，如果底层基础设施出现故障，服务调度器会重新安排任务。您可以使用任务放置策略和约束来自定义调度器的任务放置和终止方式。如果服务中的某个任务停止，调度器将启动一个新的任务来替代它。此过程将一直持续，直到服务根据服务使用的调度策略达到预期任务数。服务的调度策略也称为服务类型。

在容器运行状况检查或负载均衡器目标组运行状况检查失败后，服务计划程序还会替换被确定为运行状况不佳的任务。此替换取决于 `maximumPercent` 和 `desiredCount` 服务定义参数。如果任务被标记为运行状况不佳，则服务计划程序将首先启动替换任务。然后会发生以下情况。

- 如果替换任务的运行状况为 `HEALTHY`，则服务计划程序将停止运行不正常的任务
- 如果替换任务的运行状况为 `UNHEALTHY`，则计划程序将停止运行状况不佳的替换任务或现有的运行状况不佳任务，以使任务总数等于 `desiredCount`。

如果 `maximumPercent` 参数限制计划程序先启动替换任务，则计划程序将一次随机停止一个运行状况不佳的任务以释放容量，然后启动替换任务。启动和停止过程将继续，直到所有运行状况不佳的任务都被运行状况正常的任务所替换。替换了所有运行状况不佳的任务并且只有运行正常的任务后，如果任务

总数超过 desiredCount，则会随机停止运行状况正常的任务，直到任务总数等于 desiredCount。有关 maximumPercent 和 desiredCount 的更多信息，请参阅服务定义参数。

服务调度器包括用来限制任务反复启动失败后重新启动的频率的逻辑。如果一个任务尚未进入 RUNNING 状态便停止，则服务调度程序会开始降低启动尝试的速度，并发出服务事件消息。此行为可防止在您解决问题之前将不必要的资源用于失败的任务。服务更新后，服务调度器会恢复正常调度行为。有关更多信息，请参阅Amazon ECS 服务节流逻辑 和查看 Amazon ECS 服务事件消息。

有两种服务计划程序策略可用：

- REPLICA：副本计划策略在集群中放置和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。有关更多信息，请参阅 副本策略。
- 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。有关更多信息，请参阅 进程守护程序策略。

> **Note**
>
> Fargate 任务不支持 DAEMON 计划策略。

进程守护程序策略

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。服务调度器会评估运行任务的任务放置约束，并停止不满足放置约束的任务。使用此策略时，无需指定预期任务数、任务放置策略，也无需使用 Service Auto Scaling 策略。

Amazon ECS 为进程守护程序任务保留容器实例计算资源，包括 CPU、内存和网络接口。当您在具有其他副本服务的集群上启动进程守护程序服务时，Amazon ECS 会优先考虑进程守护程序任务。这意味着进程守护程序任务是在实例上启动的第一个任务，也是在所有副本任务停止后要停止的最后一个任务。此策略确保资源不会被待处理的副本任务使用，并且可用于进程守护程序任务。

进程守护程序服务调度器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 状态，容器实例上的进程守护程序任务会停止。此服务计划程序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。

当指定了部署配置时，则 maximumPercent 参数的值必须为 100（以百分比形式指定），如果未设置，则是使用的默认值。minimumHealthyPercent 参数的默认值为 0（以百分比形式指定）。

更改进程守护程序服务的放置约束时，您必须重新启动该服务。Amazon ECS 会动态更新预留在符合条件的实例上进程守护程序任务的资源。对于现有实例，调度器尝试将任务放在实例上。

当任务定义中的任务大小或容器资源预留变更时，会启动新部署。Amazon ECS 为守护程序选取更新后的 CPU 和内存预留，然后阻止该守护程序任务的容量。

如果没有足够的资源用于上述任一情况，则会发生以下情况：

- 任务放置失败。
- 将生成一个 CloudWatch 事件。
- Amazon ECS 通过等待资源变为可用，继续尝试并计划实例上的任务。
- Amazon ECS 释放不再符合放置约束条件的所有预留实例，并停止相应的守护进程任务。

守护进程调度策略可用于以下情况：

- 运行应用程序容器
- 运行用于记录、监视和跟踪任务的支持容器

使用 Fargate 启动类型或 CODE_DEPLOY 或 EXTERNAL 部署控制器类型的任务不支持守护程序调度策略。

当服务计划程序停止运行任务时，它将尝试在您的群集的可用区之间保持平衡。该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，如果服务已定义可用区分布策略，则会选择一项任务终止，以让剩余任务保持最佳分布。
- 如果尚未定义放置策略，请使用以下逻辑在集群中的可用区之间保持平衡：
  - 对有效容器实例进行排序。优先考虑在各自的可用区中具有此服务的最大运行任务数的实例。例如，如果 A 区有 1 个运行服务任务，B 区和 C 区各有 2 个运行服务任务，则 B 区或 C 区中的容器实例被认为最适合终止任务。
  - 基于之前的步骤，停止最佳可用区中容器实例上的任务。优先使用具有此服务的最大运行任务数的容器实例。

副本策略

副本 计划策略在集群上放置并维护所需数量的任务。

对于在 Fargate 上运行任务的服务，当服务调度器启动新任务或停止运行任务时，服务调度器会尽力尝试在可用区之间保持平衡。无需指定任务放置策略或约束。

创建在 EC2 实例上运行任务的服务时，您还可选择指定任务放置策略和约束来自定义任务放置决策。如果未指定任务放置策略或约束，则默认情况下，服务调度器会在可用区之间分布任务。该调度器使用以下逻辑：

- 确定集群中的哪些容器实例可支持服务的任务定义（例如，所需的 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。
- 如果您的副本服务依赖于进程守护程序服务（例如，在任务可以使用日志记录之前需要运行的进程守护进程日志路由器任务），请创建一个任务放置约束，确保进程守护程序服务任务在副本服务任务开始之前放置在 EC2 实例上。有关更多信息，请参阅 Amazon ECS 任务放置约束示例。
- 若已定义放置策略，请使用该策略，从剩余候选项中选择一个实例。
- 如果尚未定义放置策略，请使用以下逻辑来平衡集群中各个可用区的任务：
  - 对有效容器实例进行排序。优先考虑在各自的可用区中具有此服务的最小运行任务数的实例。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。
  - 基于之前的步骤，将新的服务任务放置在最佳可用区中的有效容器实例上。优先使用具有此服务的最小运行任务数的容器实例。

Amazon ECS 服务参数的最佳实践

为确保应用程序不会停机，按照如下部署过程操作：

1. 启动新的应用程序容器，同时保持现有容器运行。
2. 检查新容器是否正常运行。
3. 停止旧容器。

根据您的部署配置和集群中空闲的未预留空间量，可能需要多轮操作完成此操作，才能将所有旧任务替换为新任务。

您可以使用两个 ECS 服务配置选项修改数字：

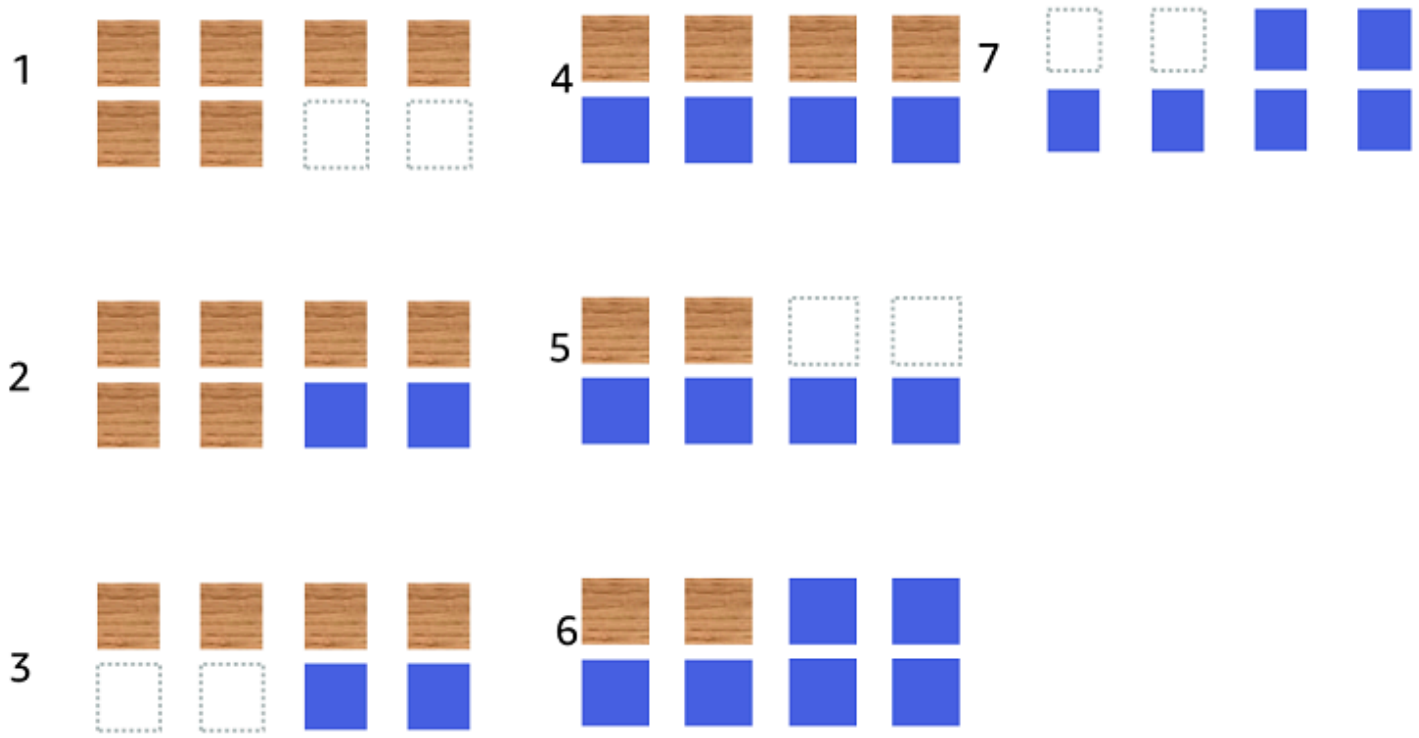
- `minimumHealthyPercent`：100%（默认值）

服务在部署期间必须保持在 RUNNING 状态的任务数下限。其表示为向上取整到最近整数的 desiredCount 的百分比。此参数使您不必使用额外的集群容量就能部署。

- maximumPercent：200%（默认值）

  服务在部署期间允许处于 RUNNING 或 PENDING 状态的任务数上限。其表示为向下取整到最近整数的 desiredCount 的百分比。

考虑以下服务，它有 6 个棕褐色任务，部署在一个拥有总共可容纳 8 个任务的空间的集群中。默认的 Amazon ECS 服务配置选项不允许部署下降到低于六个所需任务的 100%。



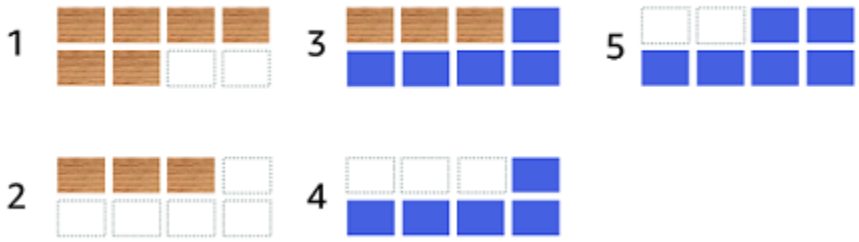
部署过程如下：

1. 目标是将棕褐色任务替换为蓝色任务。
2. 计划程序会启动两个新的蓝色任务，因为默认设置要求有六个正在运行的任务。
3. 计划程度会停止其中两个棕褐色任务，因为总共将有六个任务（四个棕褐色和两个蓝色）。
4. 计划程序会启动另外两个蓝色任务。
5. 计划程序关闭了两个棕褐色任务。
6. 计划程序会启动另外两个蓝色任务。
7. 计划程序关闭了最后两个棕褐色任务。

在上面的示例中，如果您使用选项的默认值，则每启动一个新任务都有 2.5 分钟的等待时间。此外，负载均衡器可能需要等待 5 分钟才能停止旧任务。

您可以通过将 `minimumHealthyPercent` 值设置为 50% 来加快部署速度。

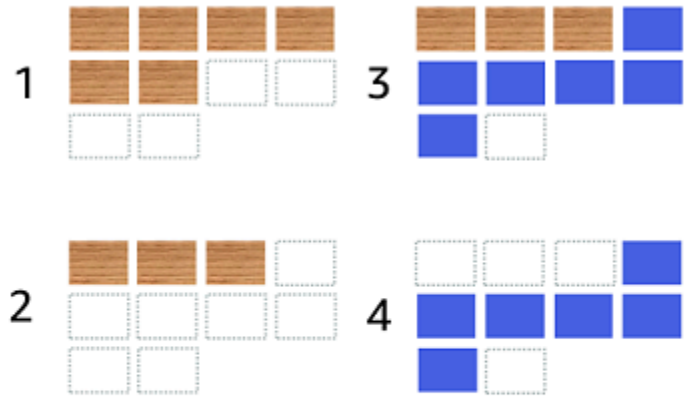
考虑以下服务，它有 6 个棕褐色任务，部署在一个拥有总共可容纳 8 个任务的空间的集群中。



部署过程如下：

1. 目标是将棕褐色任务替换为蓝色任务。
2. 计划程序会停止其中三个棕褐色任务。仍然有三个符合 `minimumHealthyPercent` 值的棕褐色任务在运行。
3. 计划程序启动五个蓝色任务。
4. 计划程序会停止剩余的三个棕褐色任务。
5. 计划程序启动最后的蓝色任务。

您还可以添加额外的可用空间，以便可以运行额外的任务。



部署过程如下：

1. 目标是将棕褐色任务替换为蓝色任务。
2. 计划程序会停止其中三项棕褐色任务
3. 计划程序启动六个蓝色任务
4. 计划程序会停止三个棕褐色任务。

当您的任务闲置一段时间并且利用率不高时，请使用以下值作为 Amazon ECS 服务配置选项。

- `minimumHealthyPercent`：50%
- `maximumPercent`：200%

使用控制台创建 Amazon ECS 服务

您可以使用控制台创建服务。

使用控制台时请考虑以下事项：

- 有两个计算选项可分发您的任务。

  - capacity provider strategy（容量提供程序策略）会使 Amazon ECS 在一个或多个容量提供商之间分发您的任务。
  - 启动类型会让 Amazon ECS 直接在 Fargate 或注册到集群的 Amazon EC2 实例上启动我们的任务。
- 任务定义使用 `awsvpc` 网络模式或配置为使用负载均衡器的服务必须具有网络配置。预设情况下，控制台会选择原定设置 Amazon VPC 以及原定设置 Amazon VPC 中的所有子网和原定设置安全组。
- 默认的任务放置策略会将任务平均分配到各可用区。
- 当您为服务部署使用 Launch Type（启动类型）时，默认情况下，该服务会在集群 VPC 的子网中启动。
- 对于 capacity provider strategy（容量提供程序策略），控制台会默认选择一个计算选项。下面介绍了控制台用于选择原定设置值的顺序：
  - 如果您的集群已定义原定设置容量提供程序策略，则将选择该策略。
  - 如果您的集群没有定义的原定设置容量提供程序策略，但您确实已将 Fargate 容量提供程序添加到集群中，则选择使用 `FARGATE` 容量提供程序的自定义容量提供程序策略。
  - 如果您的群集没有定义原定设置容量提供程序策略，但您已将一个或多个自动扩缩组容量提供程序添加到集群中，则选中使用自定义（高级）选项，且您需要手动定义策略。
  - 如果您的集群没有定义原定设置容量提供程序策略，并且没有向集群添加容量提供程序，则会选择 Fargate 启动类型。
- 默认的部署故障检测默认选项是使用 Amazon ECS 部署断路器选项和故障时回滚选项。

  有关更多信息，请参阅 Amazon ECS 部署断路器如何检测故障。
- 如果要使用蓝绿部署选项，请确定 CodeDeploy 如何移动应用程序。以下选项可用：
  - CodeDeployDefault.ECSAllAtOnce：将所有流量一次性转移到更新后的 Amazon ECS 容器
  - CodeDeployDefault.ECSLinear10PercentEvery1Minutes：在所有流量转移之前，每分钟转移 10% 的流量。
  - CodeDeployDefault.ECSLinear10PercentEvery3Minutes：在所有流量转移之前，每 3 分钟转移 10% 的流量。
  - CodeDeployDefault.ECSCanary10Percent5Minutes：在第一个增量中转移 10% 的流量。其余 90% 部署在五分钟后进行转移。
  - CodeDeployDefault.ECSCanary10Percent15Minutes：在第一个增量中转移 10% 的流量。其余 90% 部署在 15 分钟后进行转移。

- 如果您需要一个应用程序来连接到在 Amazon ECS 中运行的其他应用程序，则请确定适合您的架构的选项。有关更多信息，请参阅 互连 Amazon ECS 服务。
- 您必须使用 AWS CloudFormation 或 AWS Command Line Interface 来部署使用以下任一参数的服务：
  - 使用自定义指标跟踪策略
  - 更新服务 – 您无法更新 awsvpc 网络配置和运行状况检查宽限期。

  有关如何使用 AWS CLI 创建服务的信息，请参阅《AWS Command Line Interface 参考》中的 create-service。

  有关如何使用 AWS CloudFormation 创建服务的信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::ECS::Service。

快速创建服务

您可以使用控制台快速创建和部署服务。服务具有以下配置：

- 在与您的集群关联的 VPC 和子网中部署
- 部署一项任务
- 使用滚动部署
- 将容量提供程序策略与您的默认容量提供程序一起使用
- 使用部署断路器检测故障，并将选项设置为在出现故障时自动回滚部署

要使用默认参数部署服务，请执行以下步骤。

创建服务（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航页面中，选择集群。
3. 在集群页面上，选择要在其中创建服务的集群。
4. 在 Services（服务）选项卡上，选择 Create（创建）。
5. 在 Deployment configuration（部署配置）下，指定应用程序的部署方式。
   a. 对于 Application type（应用程序类型），选择 Service（服务）。
   b. 对于任务定义，选择要使用的任务定义族和修订。

c. 对于 Service name（服务名称），为您的服务输入一个名称。

d. 对于 Desired tasks（预期任务），输入要在服务中启动并保留的任务数量。

6. （可选）为了帮助确定您的服务和任务，请展开 Tags（标签）部分，然后配置您的标签。

   要让 Amazon ECS 使用集群名称和任务定义标签自动标记全部新启动的任务，选择 Turn on Amazon ECS managed tags（启用 Amazon ECS 托管标签），然后选择 Task definitions（任务定义）。

   要让 Amazon ECS 使用集群名称和服务标签自动标记全部新启动的任务，选择 Turn on Amazon ECS managed tags（启用 Amazon ECS 托管标签），然后选择 Service（服务）。

   添加或删除标签。

   - [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
     - 对于 Key（键），输入键名称。
     - 对于值，输入键值。
   - [删除标签] 在标签旁，选择 Remove tag (删除标签)。

使用定义的参数创建服务

要使用定义的参数创建服务，请执行以下步骤。

创建服务（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 确定要从其中启动服务的资源。

从中启动服务	步骤
集群	a. 在集群页面上，选择要在其中创建服务的集群。 b. 在 Services（服务）选项卡上，选择 Create（创建）。

从中启动服务	步骤
启动类型	a. 在任务定义页面上，选择任务定义旁边的选项按钮。 b. 在部署菜单上，选择创建服务。

3. （可选）选择任务在集群基础设施中的分发方式。展开 Compute configuration（计算配置），然后选择您的选项。

分配方式	步骤
容量提供程序策略	a. 在计算选项下，选择容量提供程序策略。 b. 选择策略： • 要使用集群的原定设置容量提供程序策略，请选择 Use cluster default（使用集群原定设置）。 • 如果集群没有默认的容量提供程序策略，或者要使用自定义策略，请选择使用自定义，添加容量提供程序策略，然后通过指定基准、容量提供程序和权重来定义自定义容量提供程序策略。 **Note** 要在策略中使用容量提供程序，容量提供

分配方式	步骤
	程序必须与集群相关联。
启动类型	a. 在 Compute options（计算选项）部分中，选择 Launch type（启动类型）。 b. 对于 Launch type（启动类型），请选择一种启动类型。 c. （可选）当指定 Fargate 启动类型时，对于平台版本中，指定要使用的平台版本。如果未指定任何版本，将使用 LATEST 平台版本。

4. 要指定服务的部署方式，请转到部署配置部分，然后选择您的选项。

   a. 对于应用程序类型，仍选择服务。

   b. 对于 Task definition（任务定义）和 Revision（修订），选择要使用的任务定义系列和修订。

   c. 对于 Service name（服务名称），为您的服务输入一个名称。

   d. 对于 Service type（服务类型），选择服务计划策略。

      - 要让计划程序只在每个活动容器实例上部署一个任务，以满足所有任务放置约束，选择 Daemon（进程守护程序）。
      - 要让计划程序在集群上放置并维护所需数量的任务，选择 Replica（副本）。

   e. 如果您对于 Desired tasks（预期任务）使用 Replica（副本），输入要在服务中启动并保留的任务数量。

   f. 确定服务的部署类型。展开部署选项，然后指定以下参数。

Deployment type（部署类型）	步骤	
滚动更新	a. 对于Min running tasks（最小运行任务数），输入服务中在部署期间必须保持 RUNNING 状态的任务数的下限，以所需任务数的百分比表示（四舍五入到最接近的整数）。有关更多信息，请参阅部署配置。 b. 对于 Max running tasks（最大运行任务数），输入部署期间 RUNNING 或 PENDING 状态下允许的服务中任务数的上限，以所需任务数的百分比表示（四舍五入到最接近的整数）。	
蓝绿部署	a. 对于部署配置，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。 b. 对于 CodeDeploy 的服务角色，请选择该服务用于向授权的 AWS 服务 发出 API 请求的 IAM 角色。	

g. 要配置 Amazon ECS 如何检测和处理部署故障，请展开 Deployment failure detection（部署故障检测），然后选择您的选项。

   i. 要在任务无法启动时停止部署，请选择 Use the Amazon ECS deployment circuit breaker（使用 Amazon ECS 部署断路器）。

要让软件在部署断路器将部署设置为故障状态时自动将部署回滚到上次完成的部署状态，请选择故障时回滚。

ii. 要根据应用程序指标停止部署，请选择使用 CloudWatch 警报。然后，从 CloudWatch 警报名称中选择警报。要创建新报警，请转到 CloudWatch 控制台。

要让软件在 CloudWatch 警报将部署设置为故障状态时自动将部署回滚到上次完成的部署状态，请选择故障时回滚。

5. （可选）要使用 Service Connect，请选择 Turn on Service Connect（开启 Service Connect），然后指定以下内容：

a. 在 Service Connect configuration（Service Connect 配置）下，指定客户端模式。

- 如果您的服务运行的网络客户端应用程序只需要连接到命名空间中的其他服务，请选择仅限客户端。
- 如果您的服务运行网络或 Web 服务应用程序且需要为该服务提供端点并连接到命名空间中的其他服务，请选择 Client and server（客户端和服务器）。

b. 要使用默认集群命名空间以外的命名空间，对于 Namespace（命名空间），请选择服务命名空间。

c. （可选）选择 Use log collection（使用日志收集）选项来指定日志配置。对于每个可用的日志驱动程序，都有日志驱动程序选项要指定。默认选项将容器日志发送到 CloudWatch Logs。其他日志驱动程序选项都使用 AWS FireLens 进行配置。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。

下面更详细地介绍了每个容器日志目标。

- Amazon CloudWatch – 将任务配置为将容器日志发送到 CloudWatch Logs。提供了默认的日志驱动程序选项，用于代表您创建 CloudWatch 日志组。要指定其他日志组名称，请更改驱动程序选项值。
- Amazon Data Firehose – 将任务配置为将容器日志发送到 Firehose。提供了默认的日志驱动程序选项，这些选项将日志发送到 Firehose 传输流。要指定其他传输流名称，请更改驱动程序选项值。
- Amazon Kinesis Data Streams – 将任务配置为将容器日志发送到 Kinesis Data Streams。提供了默认的日志驱动程序选项，这些选项将日志发送到 Kinesis Data Streams 流。要指定其他传输流名称，请更改驱动程序选项值。
- Amazon OpenSearch Service – 将任务配置为将容器日志发送到 OpenSearch Service 域。必须提供日志驱动程序选项。

- Amazon S3 – 将任务配置为将容器日志发送到 Amazon S3 存储桶。提供了默认的日志驱动程序选项，但您必须指定有效的 Amazon S3 存储桶名称。

6. （可选）要使用服务发现，请选择使用服务发现，然后指定以下内容。

   a. 要使用新的命名空间，请在配置命名空间下选择创建新命名空间，然后提供命名空间名称和描述。要使用现有命名空间，请选择选择现有命名空间，然后选择要使用的命名空间。

   b. 提供服务发现的服务信息，例如服务的名称和描述。

   c. 要让 Amazon ECS 定期执行容器级别的运行状况检查，请选择启用 Amazon ECS 任务运行状况传播。

   d. 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，具体取决于您的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 Amazon Route 53开发者指南中的支持的 DNS 记录类型。

      - 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。如果您选择 A，请跳到下一步。如果您选择 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。

      对于 TTL，请输入 DNS 解析器和 Web 浏览器缓存记录集的时长（以秒为单位）。

7. （可选）要为您的服务配置负载均衡器，请展开 Load balancing（负载均衡）。

   选择负载均衡器。

使用此负载均衡器	请执行该操作
应用程序负载均衡器	a. 对于 负载均衡器类型，选择 Application Load Balancer。 b. 选择创建新负载均衡器来创建新的 Application Load Balancer，或使用现有负载均衡器选择现有 Application Load Balancer。

使用此负载均衡器	请执行该操作	
	c. 对于 Load balancer name（负载均衡器名称），输入唯一的名称。 d. 对于 Choose container to load balance（选择用于负载均衡的容器），选择托管服务的容器。 e. 对于 Listener（侦听器），为应用程序负载均衡器输入一个端口和协议，以侦听连接请求。预设情况下，负载均衡器将配置为使用端口 80 和 HTTP。 f. 对于 Target group name（目标组名称），输入应用程序负载均衡器要将请求路由到的目标组的名称和协议。默认情况下，目标组会将请求路由到任务定义中定义的第一个容器。 g. 对于注销延迟，请输入负载均衡器将目标状态更改为 UNUSED 的秒数。默认值为 300 秒。 h. 对于 Health check path（运行状况检查路径），输入容器中存在的路径，应用程序负载均衡器应定期发送请求以验证应用程序负载均衡器和容器之间的连接运行状况。默认路径为根目录（/）。	

使用此负载均衡器	请执行该操作	
	i. 对于 Health check grace period（运行状况检查宽限期），输入服务计划程序忽略未正常运行的 Elastic Load Balancing 目标运行状况检查的时长（以秒为单位）。	

使用此负载均衡器	请执行该操作
网络负载均衡器	a. 对于 Load balancer type（负载均衡器类型），选择网络负载均衡器。 b. 对于 Load Balancer（负载均衡器），选择一个现有的网络负载均衡器。 c. 对于 Choose container to load balance（选择用于负载均衡的容器），选择托管服务的容器。 d. 对于 Target group name（目标组名称），输入网络负载均衡器要将请求路由到的目标组的名称和协议。默认情况下，目标组会将请求路由到任务定义中定义的第一个容器。 e. 对于注销延迟，请输入负载均衡器将目标状态更改为 UNUSED 的秒数。默认值为 300 秒。 f. 对于 Health check path（运行状况检查路径），输入容器中存在的路径，网络负载均衡器应定期发送请求以验证应用程序负载均衡器和容器之间的连接运行状况。默认路径为根目录（/）。 g. 对于 Health check grace period（运行状况检查宽限期），输入服务计划程序

使用此负载均衡器	请执行该操作	
	忽略未正常运行的 Elastic Load Balancing 目标运行状况检查的时长（以秒为单位）。	

8. （可选）要配置服务自动扩缩，请展开服务自动扩缩，然后指定以下参数。

   a. 要使用服务自动扩缩，请选择 Service auto scaling（服务自动扩缩）。

   b. 对于最小任务数，输入供服务自动扩缩使用的任务数的下限。所需计数不会低于此计数。

   c. 对于最大任务数，输入供服务自动扩缩使用的任务数的上限。所需计数不会高于此计数。

   d. 选择策略类型。在扩展策略类型下，选择以下选项之一。

要使用该策略，请键入...	请执行此操作...	
目标跟踪	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（策略名称），请输入策略的名称。 c. 对于 ECS 服务指标，选择以下指标之一。 • ECSServiceAverageCPUUtilization – 服务的平均 CPU 使用率。 • ECSServiceAverageMemoryUtilization – 服务的平均内存使用率。 • ALBRequestCountPerTarget – 应用程序负载均衡器目标组中每个目标完成的请求数。	

要使用该策略，请键入...	请执行此操作...	
	d. 对于 Target value（目标值），输入服务为所选指标保留的值。 e. 对于横向扩展冷却时间，输入一个横向扩展活动（添加任务）结束后，在另一个横向扩展活动启动之前必须经过的时间量（以秒为单位）。 f. 对于横向缩减冷却时间，输入一个横向缩减活动（删除任务）结束后，在另一个横向缩减活动启动之前必须经过的时间量（以秒为单位）。 g. 要防止策略执行缩减活动，请选择 Turn off scale-in（关闭横向缩减）。 h. •（可选）如果您希望扩展策略针对增加的流量进行横向扩展，但不需要在流量减少时横向缩减，请选择关闭横向缩减。	

要使用该策略，请键入...	请执行此操作...	
分步扩缩	a. 对于 Scaling policy type (扩展策略类型) 选择 Step scaling (分步扩展)。 b. 对于策略名称，请输入策略的名称。 c. 对于 Alarm name (警报名称)，为警报输入一个唯一名称。 d. 对于 Amazon ECS 服务指标，选择要用于警报的指标。 e. 对于统计信息，请选择警报统计信息。 f. 对于周期，选择警报的周期。 g. 对于警报条件，请选择如何将所选指标与定义的阈值进行比较。 h. 在用于比较指标的阈值和启动警报的评估周期中，请输入用于警报的阈值和评估阈值的时长。 i. 在扩展操作下，执行以下操作： • 对于操作，选择是否为您的服务增加、删除或设置具体预期数量。 • 如果您选择增加或删除任务，对于值，请输入启动扩展操作后要增加或删除的任务数（或	

要使用该策略，请键入...	请执行此操作...	
	现有任务的百分比）。如果您选择设置了所需的数量，则请输入任务数量。在类型中，选择值是整数还是现有所需计数的百分比值。 • 对于下限和上限，输入分步扩展调整的下限和上限。默认情况下，添加策略的下限为警报阈值，上限为正 (+) 无穷。默认情况下，移除策略的上限为警报阈值，下限为负 (-) 无穷。 • （可选）添加其他扩展选项。选择添加新的扩展操作，然后重复扩展操作步骤。 • 对于冷却时间，输入等待先前的扩展活动生效的时间（以秒为单位）。对于添加策略，该时间是在横向扩展活动之后，扩展策略阻止横向缩减活动并限制一次可以横向扩展的任务数量的时间。对于移除策略，该时间是横向缩减活动结束后，在另一个横向缩减活动可以开始前经过的时间。	

9. （可选）要使用默认策略之外的其他任务放置策略，请展开 Task Placement（任务放置），然后从以下选项中进行选择。

有关更多信息，请参阅 Amazon ECS 如何将任务放置在容器实例上。

- 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 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。
- 装填 – 根据 CPU 或内存的最低可用量来分配任务。
- 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。
- 自定义 – 定义您自己的任务放置策略。

如果您选择了 Custom（自定义），请定义放置任务的算法和任务放置过程中要考虑的规则。

- 在 Strategy（策略）下，对于 Type（类型）和 Field（字段），选择算法和用于该算法的实体。

  您最多可输入 5 个策略。
- 在约束下，对于类型和表达式，选择要用于该约束的规则和属性。

  例如，要设置在 T2 实例上放置任务的约束，对于 Expression（表达式），输入 attribute:ecs.instance-type =~ t2.*。

  您最多可输入 10 个约束。

10. 如果您的任务定义使用 awsvpc 网络模式，请展开 Networking（联网）。使用以下步骤指定自定义配置。

    a. 对于 VPC，选择要使用的 VPC。

    b. 对于 Subnets（子网），选择 VPC 中的一个或多个子网，任务计划程序在放置任务时会考虑这些子网。

    > **Important**
    >
    > awsvpc 网络模式仅支持私有子网。任务不接收公有 IP 地址。因此，出站互联网访问需要 NAT 网关，且入站互联网流量通过负载均衡器进行路由。

    c. 对于安全组，您可以选择现有安全组或创建新安全组。要使用现有安全组，请选择该安全组并移至下一步。要创建新安全组，请选择 Create a new security group (创建新安全组)。您必须指定安全组名称、说明，然后为该安全组添加一个或多个入站规则。

11. 如果您的任务使用的数据卷与部署时的配置兼容，则可以通过扩展卷来配置该卷。

卷名称和卷类型在您创建任务定义修订时配置，在创建服务时无法更改。要更新卷名称和类型，您必须创建新的任务定义修订并使用新修订创建服务。

要配置此卷类型	请执行该操作
Amazon EBS	a. 为 EBS 卷类型选择要附加到任务的 EBS 卷的类型。 b. 为大小（GiB）输入卷大小的有效值，以吉字节（GiB）为单位。您可以指定最小 1GiB 和最大 16384GiB 的卷大小。除非您提供快照 ID，否则此值为必填值。 c. 对于 IOPS，输入该卷应提供的输入/输出操作的次数（IOPS）的最大值。此值只能针对 `io1`、`io2` 和 gp3 卷类型进行配置。 d. 对于吞吐量（MiB/s），以每秒兆字节（MiBps 或 MiB/s）为单位输入卷应提供的吞吐量。此值只能针对 gp3 卷类型进行配置。 e. 对于快照 ID，请选择现有的 Amazon EBS 卷快照，或者如果您想从快照中创建卷，请输入快照的 ARN。您还可以通过不选择或输入快照 ID 来创建新的空卷。 f. 对于文件系统类型，请选择将用于在卷上数据存储和检索的文件系统的类型。您可以选择操作系统默认类型或特定的文件系统类型。Linux 的默认设置

要配置此卷类型	请执行该操作	
	为 XFS。对于从快照创建的卷，必须指定创建快照时卷使用的相同文件系统类型。如果文件系统类型不匹配，则任务将无法启动。 g. 对于基础设施角色，请选择一个具有必要权限的 IAM 角色，以允许 Amazon ECS 管理任务的 Amazon EBS 卷。您可以将 `AmazonECSInfrastructureRolePolicyForVolumes` 托管策略附加到该角色，也可以使用策略作为指南，以创建并附加您自己的具有满足您特定需求的权限的策略。有关必要权限的更多信息，请参阅Amazon ECS 基础设施 IAM 角色。 h. 对于加密，如果您想使用 Amazon EBS 默认加密设置，则请选择默认。如果您的账户配置了默认加密，则该卷将使用设置中指定的 AWS Key Management Service（AWS KMS）密钥进行加密。如果您选择默认，但未开启 Amazon EBS 默认	

要配置此卷类型	请执行该操作	
	加密，则该卷将处于未加密状态。 如果选择自定义，则可以为卷加密指定您选择的 AWS KMS key。 如果选择无，则除非您配置了加密，或者您使用加密的快照创建了卷，否则该卷将处于未加密状态。 i. 如果您选择了自定义作为加密，则必须指定要使用的 AWS KMS key。为 KMS 密钥选择 AWS KMS key 或输入密钥 ARN。如果您选择使用对称的客户托管密钥对卷进行加密，请确保您拥有 AWS KMS key 策略中定义的正确权限。有关更多信息，请参阅 Amazon EBS 卷的数据加密。 j. （可选）在标签下，您可以通过传播任务定义或服务中的标签或提供自己的标签来将标签添加到 Amazon EBS 卷中。 如果您要传播任务定义中的标签，请为从中传播标签选择任务定义。如果要从服务中传播标签，则请为传播标签来源选择服务。如果您选择不传播，	

要配置此卷类型	请执行该操作	
	或者如果您未选择值，则不会传播标签。 如果您想提供自己的标签，请选择添加标签，然后为您添加的每个标签提供密钥和值。 有关标记 Amazon EBS 卷的更多信息，请参阅标记 Amazon EBS 卷。	

12. （可选）为了帮助确定您的服务和任务，请展开 Tags（标签）部分，然后配置您的标签。

    要让 Amazon ECS 使用集群名称和任务定义标签自动标记全部新启动的任务，选择开启 Amazon ECS 托管标签，然后对于从中传播标签，选择任务定义。

    要让 Amazon ECS 使用集群名称和服务标签自动标记全部新启动的任务，选择开启 Amazon ECS 托管标签，然后对于从中传播标签，选择服务。

    添加或删除标签。

    - [添加标签] 选择 Add tag（添加标签），然后执行以下操作：
      - 对于 Key（键），输入键名称。
      - 对于值，输入键值。
    - [删除标签] 在标签旁，选择 Remove tag (删除标签)。

使用控制台更新 Amazon ECS 服务

您可以使用 Amazon ECS 控制台更新 Amazon ECS 服务。当前服务配置已预先填充。您可以更新任务定义、所需任务计数、容量提供程序策略、平台版本和部署配置；或者这些的任意组合。

有关如何更新蓝绿部署配置的信息，请参阅 使用控制台更新 Amazon ECS 蓝绿部署。

使用控制台时请考虑以下事项：

如果要暂时停止服务，则请将所需任务设置为 0。然后，当您准备好启动服务时，请使用最初的所需任务计数更新该服务。

使用控制台时请考虑以下事项：

- 您必须使用 AWS Command Line Interface 来更新使用以下任一参数的服务：
  - 蓝/绿部署
  - 服务发现 – 您只能查看您的服务发现配置。
  - 使用自定义指标跟踪策略
  - 更新服务 – 您无法更新 awsvpc 网络配置和运行状况检查宽限期。

  有关如何使用 AWS CLI 更新服务的信息，请参阅《AWS Command Line Interface 参考》中的 update-service。
- 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的安全组以使用更新后的端口。
- 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。
- 如果您的服务使用一个负载均衡器，则无法使用控制台更改创建该负载均衡器时为服务定义的负载均衡器配置。您可以改而使用 AWS CLI 或开发工具包来修改负载均衡器配置。有关如何修改配置的信息，请参阅《Amazon Elastic Container Service API 参考》中的 UpdateService。
- 如果您更新服务的任务定义，则在负载均衡器中指定的容器名称和容器端口必须保留在任务定义中。

您可以更新现有服务以更改一些服务配置参数，如该服务维护的任务数、任务使用哪个任务定义，或者，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。使用 Linux 平台版本的服务无法更新为使用 Windows 平台版本，反之亦然。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。

如果要为任务使用更新的容器映像，则可以使用该映像创建新的任务定义修订版本，并使用控制台中的 force new deployment (强制新部署) 选项将其部署到服务。

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。可以使用此参数进行部署，而无需使用额外的集群容量。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，则最大百分比参数将以预期任务数的百分比形式（向下取整到最近的整数）表示部署期间允许处于 PENDING、RUNNING 或 STOPPING 状态的服务中任务数的上限。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。可以使用此参数定义部署批次大小。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%。

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。

在容器运行状况检查或负载均衡器目标组运行状况检查失败后，服务计划程序还会替换被确定为运行状况不佳的任务。此替换取决于 maximumPercent 和 desiredCount 服务定义参数。如果任务被标记为运行状况不佳，则服务计划程序将首先启动替换任务。然后会发生以下情况。

- 如果替换任务的运行状况为 HEALTHY，则服务计划程序将停止运行不正常的任务
- 如果替换任务的运行状况为 UNHEALTHY，则计划程序将停止运行状况不佳的替换任务或现有的运行状况不佳任务，以使任务总数等于 desiredCount。

如果 maximumPercent 参数限制计划程序先启动替换任务，则计划程序将一次随机停止一个运行状况不佳的任务以释放容量，然后启动替换任务。启动和停止过程将继续，直到所有运行状况不佳的任务都被运行状况正常的任务所替换。替换了所有运行状况不佳的任务并且只有运行正常的任务后，如果任务总数超过 desiredCount，则会随机停止运行状况正常的任务，直到任务总数等于 desiredCount。有关 maximumPercent 和 desiredCount 的更多信息，请参阅服务定义参数。

> **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的安全组以使用更新后的端口。
>
> 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。
>
> 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。

更新服务（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在 Clusters（集群）页面上，选择集群。
3. 在集群详细信息页面上的服务部分，选中服务旁边的复选框，然后选择更新。
4. 要让您的服务开始新的部署，请选择 Force new deployment（强制执行新部署）。
5. 对于任务定义，选择任务定义系列和修订。

   > **Important**
   >
   > 控制台验证所选任务定义系列和修订与定义的计算配置兼容。如果收到警告，请验证任务定义兼容性和您选择的计算配置。

6. 对于预期任务，请输入要为服务运行的任务数量。
7. 对于Min running tasks（最小运行任务数），输入服务中在部署期间必须保持 RUNNING 状态的任务数的下限，以所需任务数的百分比表示（四舍五入到最接近的整数）。有关更多信息，请参阅部署配置。
8. 对于 Max running tasks（最大运行任务数），输入部署期间 RUNNING 或 PENDING 状态下允许的服务中任务数的上限，以所需任务数的百分比表示（四舍五入到最接近的整数）。
9. 要配置 Amazon ECS 如何检测和处理部署故障，请展开 Deployment failure detection（部署故障检测），然后选择您的选项。

   a. 要在任务无法启动时停止部署，请选择 Use the Amazon ECS deployment circuit breaker（使用 Amazon ECS 部署断路器）。

      要让软件在部署断路器将部署设置为故障状态时自动将部署回滚到上次完成的部署状态，请选择故障时回滚。

   b. 要根据应用程序指标停止部署，请选择使用 CloudWatch 警报。然后，从 CloudWatch 警报名称中选择警报。要创建新报警，请转到 CloudWatch 控制台。

      要让软件在 CloudWatch 警报将部署设置为故障状态时自动将部署回滚到上次完成的部署状态，请选择故障时回滚。

10. 要更改计算选项，请展开计算配置，然后执行以下操作：

    a. 对于 AWS Fargate 上的服务，请在 Platform version（平台版本）中选择新版本。

    b. 对于使用容量提供程序策略的服务，请对容量提供程序策略执行以下操作：

- 要添加其他容量提供程序，请选择添加更多。然后，对于容量提供程序，请选择容量提供程序。
- 要移除容量提供程序，请在容量提供商程序的右侧选择删除。

使用自动扩缩组容量提供程序的服务无法更新为使用 Fargate 容量提供程序。使用 Fargate 容量提供程序的服务无法更新为使用自动扩缩组容量提供程序。

11. （可选）要配置服务自动扩缩，请展开服务自动扩缩，然后指定以下参数。

    a. 要使用服务自动扩缩，请选择 Service auto scaling（服务自动扩缩）。

    b. 对于最小任务数，输入供服务自动扩缩使用的任务数的下限。所需计数不会低于此计数。

    c. 对于最大任务数，输入供服务自动扩缩使用的任务数的上限。所需计数不会高于此计数。

    d. 选择策略类型。在扩展策略类型下，选择以下选项之一。

要使用该策略，请键入...	请执行此操作...
目标跟踪	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（策略名称），请输入策略的名称。 c. 对于 ECS 服务指标，选择以下指标之一。 • ECSServiceAverageCPUUtilization – 服务的平均 CPU 使用率。 • ECSServiceAverageMemoryUtilization – 服务的平均内存使用率。 • ALBRequestCountPerTarget – 应用程序负载均衡器目标组中每个目标完成的请求数。

要使用该策略，请键入...	请执行此操作...	
	d. 对于 Target value（目标值），输入服务为所选指标保留的值。 e. 对于横向扩展冷却时间，输入一个横向扩展活动（添加任务）结束后，在另一个横向扩展活动启动之前必须经过的时间量（以秒为单位）。 f. 对于横向缩减冷却时间，输入一个横向缩减活动（删除任务）结束后，在另一个横向缩减活动启动之前必须经过的时间量（以秒为单位）。 g. 要防止策略执行缩减活动，请选择 Turn off scale-in（关闭横向缩减）。 h. •（可选）如果您希望扩展策略针对增加的流量进行横向扩展，但不需要在流量减少时横向缩减，请选择关闭横向缩减。	

要使用该策略，请键入...	请执行此操作...	
分步扩缩	a. 对于 Scaling policy type (扩展策略类型) 选择 Step scaling (分步扩展)。 b. 对于策略名称，请输入策略的名称。 c. 对于 Alarm name (警报名称)，为警报输入一个唯一名称。 d. 对于 Amazon ECS 服务指标，选择要用于警报的指标。 e. 对于统计信息，请选择警报统计信息。 f. 对于周期，选择警报的周期。 g. 对于警报条件，请选择如何将所选指标与定义的阈值进行比较。 h. 在用于比较指标的阈值和启动警报的评估周期中，请输入用于警报的阈值和评估阈值的时长。 i. 在扩展操作下，执行以下操作： • 对于操作，选择是否为您的服务增加、删除或设置具体预期数量。 • 如果您选择增加或删除任务，对于值，请输入启动扩展操作后要增加或删除的任务数（或	

要使用该策略，请键入...	请执行此操作...	
	现有任务的百分比）。如果您选择设置了所需的数量，则请输入任务数量。在类型中，选择值是整数还是现有所需计数的百分比值。 • 对于下限和上限，输入分步扩展调整的下限和上限。默认情况下，添加策略的下限为警报阈值，上限为正 (+) 无穷。默认情况下，移除策略的上限为警报阈值，下限为负 (-) 无穷。 • （可选）添加其他扩展选项。选择添加新的扩展操作，然后重复扩展操作步骤。 • 对于冷却时间，输入等待先前的扩展活动生效的时间（以秒为单位）。对于添加策略，该时间是在横向扩展活动之后，扩展策略阻止横向缩减活动并限制一次可以横向扩展的任务数量的时间。对于移除策略，该时间是横向缩减活动结束后，在另一个横向缩减活动可以开始前经过的时间。	

12. （可选）要使用 Service Connect，请选择 Turn on Service Connect（开启 Service Connect），然后指定以下内容：

a. 在 Service Connect configuration（Service Connect 配置）下，指定客户端模式。

- 如果您的服务运行的网络客户端应用程序只需要连接到命名空间中的其他服务，请选择 Client side only（仅限客户端）。
- 如果您的服务运行网络或 Web 服务应用程序且需要为该服务提供端点并连接到命名空间中的其他服务，请选择 Client and server（客户端和服务器）。

b. 要使用默认集群命名空间以外的命名空间，对于 Namespace（命名空间），请选择服务命名空间。

13. 如果您的任务使用的数据卷与部署时的配置兼容，则可以通过扩展卷来配置该卷。

卷名称和卷类型在您创建任务定义修订时配置，在更新服务时无法更改。要更新卷名称和类型，您必须创建新的任务定义修订并使用新修订更新服务。

要配置此卷类型	请执行该操作
Amazon EBS	a. 为 EBS 卷类型选择要附加到任务的 EBS 卷的类型。 b. 为大小（GiB）输入卷大小的有效值，以吉字节（GiB）为单位。您可以指定最小 1GiB 和最大 16384GiB 的卷大小。除非您提供快照 ID，否则此值为必填值。 c. 对于 IOPS，输入该卷应提供的输入/输出操作的次数（IOPS）的最大值。此值只能针对 `io1`、`io2` 和 `gp3` 卷类型进行配置。 d. 对于吞吐量（MiB/s），以每秒兆字节（MiBps 或 MiB/s）为单位输入卷应提供的吞吐量。此值只能针对 `gp3` 卷类型进行配置。

要配置此卷类型	请执行该操作	
	e. 对于快照 ID，请选择现有的 Amazon EBS 卷快照，或者如果您想从快照中创建卷，请输入快照的 ARN。您还可以通过不选择或输入快照 ID 来创建新的空卷。 f. 对于文件系统类型，请选择将用于在卷上数据存储和检索的文件系统的类型。您可以选择操作系统默认类型或特定的文件系统类型。Linux 的默认设置为 XFS。对于从快照创建的卷，必须指定创建快照时卷使用的相同文件系统类型。如果文件系统类型不匹配，则任务将无法启动。 g. 对于基础设施角色，请选择一个具有必要权限的 IAM 角色，以允许 Amazon ECS 管理任务的 Amazon EBS 卷。您可以将 `AmazonECSInfrastructureRolePolicyForVolumes` 托管策略附加到该角色，也可以使用策略作为指南，以创建并附加您自己的具有满足您特定需求的权限的策略。有关必要权限的更多信息，请参	

要配置此卷类型	请执行该操作	
	阅Amazon ECS 基础设施 IAM 角色。 h. 对于加密，如果您想使用 Amazon EBS 默认加密设置，则请选择默认。如果您的账户配置了默认加密，则该卷将使用设置中指定的 AWS Key Management Service（AWS KMS）密钥进行加密。如果您选择默认，但未开启 Amazon EBS 默认加密，则该卷将处于未加密状态。 如果选择自定义，则可以为卷加密指定您选择的 AWS KMS key。 如果选择无，则除非您配置了加密，或者您使用加密的快照创建了卷，否则该卷将处于未加密状态。 i. 如果您选择了自定义作为加密，则必须指定要使用的 AWS KMS key。为 KMS 密钥选择 AWS KMS key 或输入密钥 ARN。如果您选择使用对称的客户托管密钥对卷进行加密，请确保您拥有 AWS KMS key 策略中定义的正确权限。有关更多信息，请参阅 Amazon EBS 卷的数据加密。	

要配置此卷类型	请执行该操作	
	j. （可选）在标签下，您可以通过传播任务定义或服务中的标签或提供自己的标签来将标签添加到 Amazon EBS 卷中。 如果您要传播任务定义中的标签，请为从中传播标签选择任务定义。如果要从服务中传播标签，则请为传播标签来源选择服务。如果您选择不传播，或者如果您未选择值，则不会传播标签。 如果您想提供自己的标签，请选择添加标签，然后为您添加的每个标签提供密钥和值。 有关标记 Amazon EBS 卷的更多信息，请参阅标记 Amazon EBS 卷。	

14. （可选）为了帮助识别您的服务，请展开 Tags（标签）部分，然后配置您的标签。
    - [添加标签] 选择添加标签，然后执行以下操作：
      - 对于 Key（键），输入键名称。
      - 对于值，输入键值。
    - [删除标签] 在标签旁，选择 Remove tag (删除标签)。
15. 选择更新。

使用控制台更新 Amazon ECS 蓝绿部署

您可以使用 Amazon ECS 控制台更新蓝绿部署配置。已预先填充当前蓝绿部署配置。您可以更新以下蓝绿部署选项：

- 部署组名称 - CodeDeploy 部署设置
- 应用程序名称 - CodeDeploy 部署组
- 部署配置 - CodeDeploy 在部署期间如何将生产流量路由到您的替换任务集
- 负载均衡器上的测试侦听器 - 在部署期间 Codeploy 使用测试侦听器将测试流量路由到替换任务集

在更新配置之前，您必须先配置新选项。

要更新蓝绿部署配置（Amazon ECS 控制台）

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在 Clusters（集群）页面上，选择集群。
3. 在 Cluster overview（集群概述）页面上，选择服务，然后选择 Update（更新）。
4. 展开部署选项 - 由 CodeDeploy 提供支持，然后选择要更新的选项：
   - 要修改 CodeDeploy 部署组，请为应用程序名称选择部署组。
   - 要修改 CodeDeploy 部署设置，请为部署组名称选择组。
   - 要修改 CodeDeploy 在部署期间如何将生产流量路由到您的替换任务集，请为部署配置选择选项。
5. 选择要作为服务部署的新修订的一部分运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。在发生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。在发生此生命周期事件时运行的 Lambda 函数结果可能触发回滚。
   - 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。在发生此生命周期事件时运行的 Lambda 函数结果可能触发回滚。
   - 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。在发生此生命周期事件时运行的 Lambda 函数结果可能触发回滚。

6. 要修改测试侦听器，请展开负载均衡，然后在 CodeDeploy 部署的测试侦听器中选择测试侦听器。
7. 选择更新。

使用控制台删除 Amazon ECS 服务

您可以使用控制台删除 Amazon ECS 服务。在删除之前，服务会自动缩减到零。与服务关联的负载均衡器资源或 Service Discovery 资源不受服务删除的影响。要删除 Elastic Load Balancing 资源，请参阅以下主题之一，具体取决于负载平衡器类型：删除 Application Load Balancer 或 删除 Network Load Balancer。

删除服务后，如果仍有需要清除的运行任务，服务状态会从 ACTIVE 转变为 DRAINING，并且服务在控制台或 ListServices API 操作中不再可见。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 状态的服务仍然可使用 DescribeServices API 操作查看。

> **Important**
>
> 如果您尝试创建与处于 ACTIVE 或 DRAINING 状态的现有服务同名的新服务，则会收到错误消息。

过程

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在 Clusters（集群）页面上，选择服务的集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 在 Cluster : ***name***（集群：名称）页面上，选择 Services（服务）选项卡。
5. 选择服务，然后选择 Delete（删除）。
6. 要删除没有缩减至零任务的服务，请选择 Force delete service（强制删除服务）。
7. 在确认提示符处，输入 delete，然后选择删除。

通过替换任务来部署 Amazon ECS 服务

当创建一项使用滚动更新（ECS）部署类型的服务时，Amazon ECS 服务调度器会将当前正在运行的任务替换为新任务。在滚动更新期间 Amazon ECS 在服务中添加或删除的任务数量由服务部署配置控制。部署配置由以下内容组成：

- `minimumHealthyPercent` 表示在部署期间或容器实例耗尽时应为服务运行的任务数的下限，服务所需任务数的百分比表示。此值四舍五入。例如，如果最小运行状况百分比为 `50`，并且所需的任务计数为 4，则调度器可以在启动两个新任务之前停止两个现有任务。同样，如果最小运行状况百分比为 75%，所需任务计数为 2，则由于结果值也为 2，调度器无法停止任何任务。

  如果任务运行不正常，Amazon ECS 服务计划程序将首先启动替换任务并维持 `minimumHealthyPercent` 任务，直到替换任务运行正常为止。替换任务启动并运行正常后，运行不正常的任务将逐渐停止。
- `maximumPercent` 表示部署期间或容器实例耗尽时应为服务运行的任务数的上限，服务所需任务数的百分比表示。此值四舍五入。例如，如果最大百分比为 `200` 且目标任务为计数四，则调度器可以在停止四个现有任务之前启动四个新任务。同样，如果最大百分比为 `125`，且目标任务计数为三，则调度器无法启动任何任务，因为结果值也是三。

> **Important**
>
> 设置最小运行状况百分比或最大百分比时，应确保调度器在启动部署时可以停止或启动至少一个任务。如果您的服务由于部署配置无效而停滞的部署，将发送服务事件消息。有关更多信息，请参阅 因为服务部署配置，服务 (*service-name*) 无法在部署期间停止或启动任务。更新 minimumHealthyPercent 或 maximumPercent 值，然后重试。。

滚动部署会使用部署断路器来确定任务是否达到稳定状态。部署断路器可以在部署失败时选择性地回滚部署。

故障检测

这两种方法可以让您快速识别部署失败的时间，然后有选择地将失败回滚到最后一个工作部署。

- the section called “ 部署断路器如何检测故障”
- the section called “CloudWatch 警报如何检测部署故障”

可以单独使用这些方法，也可以一起使用。当同时使用这两种方法时，只要满足任一故障方法的故障标准，部署就会设置为失败。

使用以下准则来帮助确定要使用哪种方法：

- 断路器 — 若要在任务无法启动时停止部署，请使用此方法。
- CloudWatch 警报 — 若要根据应用程序指标停止部署，请使用此方法。

Amazon ECS 部署断路器如何检测故障

部署断路器是一种滚动更新机制，可确定任务是否达到稳定状态。部署断路器有一个选项，该选项可自动将失败的部署回滚到处于 COMPLETED 状态的部署。

当服务部署更改状态时，Amazon ECS 会向 EventBridge 发送服务部署状态更改事件。这提供了一种用于监控服务部署状态的编程方式。有关更多信息，请参阅 Amazon ECS 服务部署状态更改事件。建议您使用 SERVICE_DEPLOYMENT_FAILED 的 eventName 创建和监控 EventBridge 规则，以便您可以采取手动操作开始您的部署。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建 EventBridge 规则

当部署断路器确定部署失败时，它会查找处于 COMPLETED 状态的最新部署。这是其用作回滚部署的部署。当回滚开始时，部署将从 COMPLETED 更改为 IN_PROGRESS。这意味着部署在达到 COMPLETED 状态之前不符合再次回滚的资格。当部署断路器找不到处于 COMPLETED 状态的部署时，断路器不会启动新任务，部署将停止。

创建服务时，调度器会跟踪分两个阶段启动失败的任务。

- 第 1 阶段 - 调度器监控任务以查看它们是否过渡到“正在运行”状态。
  - 成功 - 部署有可能过渡到“已完成”状态，因为有不止一项任务已过渡到“正在运行”状态。跳过故障标准，断路器进入第 2 阶段。
  - 失败 - 连续有任务未过渡到“正在运行”状态，部署可能会过渡到“失败”状态。
- 第 2 阶段 - 当至少有一个任务处于“正在运行”状态时，部署进入此阶段。断路器对正在评估的当前部署中的任务进行运行状况检查。经过验证的运行状况检查包括 Elastic Load Balancing、AWS Cloud Map 服务运行状况检查和容器运行状况检查。
  - 成功 - 至少有一个任务处于运行状态且运行状况检查已通过。
  - 失败 - 由于运行状况检查失败而被替换的任务已达到失败阈值。

对服务使用部署断路器方法时，请注意以下事项。EventBridge 生成规则。

- DescribeServices 响应提供了对部署状态、rolloutState 和 rolloutStateReason 的深入了解。启动新部署时，部署状态将在 IN_PROGRESS 状态开始。当服务达到稳定状态时，部署状态将转换为COMPLETED。如果服务未能达到稳定状态并打开了断路器，则部署将转换为 FAILED 状态。FAILED 状态中的部署不会启动任何新任务。
- 除了 Amazon ECS 为已启动和已完成的部署发送的服务部署状态更改事件外，Amazon ECS 还将在启用断路器的部署失败时发送事件。这些事件提供了有关部署失败的原因或部署是否由于回滚而启动的详细信息。有关更多信息，请参阅 Amazon ECS 服务部署状态更改事件。
- 如果由于以前的部署失败并出现了回滚而启动了新部署，则服务部署状态更改事件的 reason 字段将指示部署是由于回滚而启动的。
- 部署断路器仅支持使用滚动更新 (ECS) 部署控制器的 Amazon ECS 服务。
- 您必须使用 Amazon ECS 控制台，或者在使用带有 CloudWatch 选项的部署断路器时必须使用 AWS CLI。这种方法的示例是，如果有关更多信息，请参阅《AWS Command Line Interface 参考》中的 the section called “使用定义的参数创建服务” 和 create-service。

以下 create-service AWS CLI 示例显示如何在将部署断路器和回滚选项结合使用时创建 Linux 服务。

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
 \
     --task-definition sample-fargate:1 \
     --launch-type FARGATE \
     --platform-family LINUX \
     --platform-version 1.4.0 \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

例如：

部署 1 处于 COMPLETED 状态。

部署 2 无法启动，所以断路器会回滚到部署 1。部署 1 会过渡到 IN_PROGRESS 状态。

部署 3 启动，但没有处于 COMPLETED 状态的部署，因此部署 3 无法回滚或启动任务。

Failure threshold

部署断路器计算阈值，然后使用该值来确定何时将部署移动到 FAILED 状态。

部署断路器的最低阈值为 3，最大阈值为 200，并使用以下公式中的值来确定部署失败。

```
Minimum threshold <= 0.5 * desired task count => maximum threshold
```

当计算结果大于最小值 3 但小于最大值 200 时，故障阈值设置为计算的阈值（向上取整）。

> **Note**
>
> 您不能更改任何一个阈值。

部署状态检查分两个阶段。

1. 部署断路器监控部署中的任务并检查处于 RUNNING 状态的任务。调度器在当前部署中的任务位于 RUNNING 状态并进入下一阶段时忽略失败条件。当任务无法到达 RUNNING 状态时，部署断路器将失败计数增加 1。当故障计数等于阈值时，部署将标记为 FAILED。
2. 当有一个或多个任务处于 RUNNING 状态时，进入此阶段。部署断路器对当前部署中的任务的以下资源执行运行状况检查：
   - Elastic Load Balancing 负载均衡器
   - AWS Cloud Map 服务
   - Amazon ECS 容器运行状况检查

   当任务的运行状况检查失败时，部署断路器会将失败计数增加 1。当故障计数等于阈值时，部署将标记为 FAILED。

下表提供了一些示例。

预期任务计数	计算	Threshold
1	`3 <= 0.5 * 1 => 200`	3（计算值低于最低值）
25	`3 <= 0.5 * 25 => 200`	13（此值将向上舍入）

预期任务计数	计算	Threshold
400	`3 <= 0.5 * 400 => 200`	200
800	`3 <= 0.5 * 800 => 200`	200（计算值大于最高值）

例如，当阈值为 3 时，断路器在故障计数设置为 0 的情况下启动。当任务无法到达 RUNNING 状态时，部署断路器将失败计数增加 1。当故障计数等于 3 时，部署将标记为 FAILED。

有关如何使用回滚选项的其他示例，请参阅 Announcing Amazon ECS deployment circuit breaker（宣布推出 Amazon ECS 部署断路器）。

CloudWatch 警报如何检测 Amazon ECS 部署故障

您可以配置 Amazon ECS 以在检测到指定的 CloudWatch 警报已进入 ALARM 状态时，将部署设置为失败。

您可以选择设置配置，将失败的部署回滚到上一个完成的部署。

以下 `create-service` AWS CLI 示例显示如何在将部署警报和回滚选项结合使用时创建 Linux 服务。

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
     --deployment-configuration
 "alarms={alarmNames=[alarm1Name,alarm2Name],enable=true,rollback=true}" \
     --task-definition sample-fargate:1 \
     --launch-type FARGATE \
     --platform-family LINUX \
     --platform-version 1.4.0 \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

对服务使用 Amazon CloudWatch 警报方法时，请注意以下事项。

- 烘焙时间是指新服务版本横向扩展和旧服务版本横向缩减之后的一段时间，在此期间，Amazon ECS 继续监控与部署关联的警报。Amazon ECS 根据与部署关联的警报配置计算此时间段。

- `deploymentConfiguration` 请求参数现在包含 `alarms` 数据类型。您可以指定警报名称、是否使用该方法以及在警报显示部署故障时是否启动回滚。有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 CreateService。
- `DescribeServices` 响应提供了对部署状态、`rolloutState` 和 `rolloutStateReason` 的深入了解。启动新部署时，回滚状态将在 `IN_PROGRESS` 状态开始。当服务达到稳定状态时，烘焙时间完成，回滚状态将转换为 `COMPLETED`。如果服务未能达到稳定状态并且警报已进入 `ALARM` 状态，则部署将转换为 `FAILED` 状态。`FAILED` 状态中的部署不会启动任何新任务。
- 除了 Amazon ECS 为已启动和已完成的部署发送的服务部署状态更改事件外，Amazon ECS 还将在使用警报的部署失败时发送事件。这些事件提供了有关部署失败的原因或部署是否由于回滚而启动的详细信息。有关更多信息，请参阅 Amazon ECS 服务部署状态更改事件。
- 如果由于以前的部署失败并打开了回滚而启动了新部署，则服务部署状态更改事件的 `reason` 字段将指示部署是由于回滚而启动的。
- 如果您使用部署断路器和 Amazon CloudWatch 警报来检测故障，则两种方法中的任意一个都可以在满足任一方法的条件后立即启动部署失败。当您对启动部署故障的方法使用回滚选项时，会发生回滚。
- Amazon CloudWatch 警报仅支持使用滚动更新 (`ECS`) 部署控制器的 Amazon ECS 服务。
- 您可以使用新 Amazon ECS 控制台或 AWS CLI 来配置此选项。这种方法的示例是，如果有关更多信息，请参阅《AWS Command Line Interface 参考》中的 the section called “使用定义的参数创建服务” 和 create-service。
- 您可能会注意到，部署状态会长时间处于 `IN_PROGRESS`。其原因是，只有在删除活动部署后 Amazon ECS 才会更改状态，而这种情况要等到烘焙时间之后才会发生。根据您的警报配置，部署所需的时间可能比在不使用警报时长几分钟（即使新的主要任务集已纵向扩展而旧的部署已缩减）。如果您使用 CloudFormation 超时，请考虑延长超时。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的在模板中创建等待条件。
- Amazon ECS 调用 `DescribeAlarms` 轮询警报。对 `DescribeAlarms` 调用计入与您的账户相关的 CloudWatch 服务限额。如果您有调用 `DescribeAlarms` 的其他 AWS 服务，则对 Amazon ECS 的轮询警报可能有影响。例如，如果其他服务进行足够 `DescribeAlarms` 调用以达到配额，则该服务会受到限制，Amazon ECS 也会受到限制，无法轮询警报。如果在节流期内生成警报，Amazon ECS 可能会错过警报，也可能不会发生回滚。对部署没有其他影响。有关更多信息，请参阅《CloudWatch 用户指南》中的 CloudWatch 服务限额。
- 如果警报在部署开始时处于 `ALARM` 状态，则 Amazon ECS 在该部署期间不会监控警报（Amazon ECS 将忽略警报配置）。此行为解决了您想要启动新部署来修复初始部署失败的问题。

建议的警报

我们建议您使用以下警报指标：

- 如果您使用应用程序负载均衡器，请使用 HTTPCode_ELB_5XX_Count 和 HTTPCode_ELB_4XX_Count 应用程序负载均衡器指标。这些指标检查 HTTP 峰值。有关更多信息，请参阅《应用程序负载均衡器用户指南》中的应用程序负载均衡器的 CloudWatch 指标。
- 如果您有现有的应用程序，请使用 CPUUtilization 和 MemoryUtilization 指标。这些指标检查集群或服务使用的 CPU 和内存百分比。有关更多信息，请参阅 the section called “注意事项”。
- 如果您在任务中使用 Amazon Simple Queue Service 队列，请使用 ApproximateNumberOfMessagesNotVisible Amazon SQS 指标。该指标检查队列中延迟且无法立即读取的消息数量。有关更多信息，请参阅《Amazon Simple Queue Service 开发人员指南》中的 Amazon SQS 的可用 CloudWatch 指标。

在部署前验证 Amazon ECS 服务的状态

蓝/绿部署类型使用由 CodeDeploy 控制的蓝/绿部署模型。在向服务发送生产流量之前，请使用此部署类型验证服务的新部署。有关更多信息，请参阅《AWS CodeDeploy 用户指南》中的什么是 CodeDeploy。在部署前验证 Amazon ECS 服务的状态

在蓝绿部署过程中，有三种可转移流量的方法：

- Canary — 流量将通过两次递增进行转移。您可以从预定义的 Canary 选项中选择，这些选项指定在第一次增量中转移到更新后的任务集的流量百分比以及以分钟为单位的间隔；然后指定在第二次增量中转移剩余的流量。
- 线性部署 — 流量使用相等的增量转移，在每次递增之间间隔的分钟数相同。您可以从预定义的线性选项中进行选择，这些选项指定在每次增量中转移的流量百分比以及每次增量之间的分钟数。
- 一次性部署 – 所有流量均从原始任务集一次性地转移到更新后的任务集。

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 设置

有关更多信息，请参阅 AWS CodeDeploy用户手册中的使用部署组。

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。提供了以下预定义的线性和 Canary 部署配置。您还可以创建自定义的线性和 Canary 部署。有关详细信息，请参见AWS CodeDeploy 用户手册中的使用部署配置。

- CodeDeployDefault.ECSAllAtOnce：将所有流量一次性转移到更新后的 Amazon ECS 容器
- CodeDeployDefault.ECSLinear10PercentEvery1Minutes：在所有流量转移之前，每分钟转移 10% 的流量。
- CodeDeployDefault.ECSLinear10PercentEvery3Minutes：在所有流量转移之前，每 3 分钟转移 10% 的流量。
- CodeDeployDefault.ECSCanary10Percent5Minutes：在第一个增量中转移 10% 的流量。其余 90% 部署在五分钟后进行转移。
- CodeDeployDefault.ECSCanary10Percent15Minutes：在第一个增量中转移 10% 的流量。其余 90% 部署在 15 分钟后进行转移。

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 用户指南中的 AppSpec 文件参考。

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。
- 您必须将服务配置为使用 Application Load Balancer 或 Network Load Balancer。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 Network Load Balancer 时，仅支持 `CodeDeployDefault.ECSAllAtOnce` 部署配置。
- 对于配置为使用服务自动缩放和蓝色/绿色部署类型的服务，在部署期间不会阻止自动缩放，但在某些情况下，部署可能会失败。下面将更详细地介绍此行为。
  - 如果服务正在扩展且部署开始，将创建绿色任务集，CodeDeploy 将等待一个小时，让绿色任务集达到稳定状态，在达到稳定状态之前不会转移任何流量。
  - 如果服务处于蓝色/绿色部署过程中，发生了扩展事件，则流量将继续移动 5 分钟。如果服务在 5 分钟内未达到稳定状态，CodeDeploy 将停止部署并将其标记为失败。
  - 如果服务处于蓝/绿部署过程中，并发生了扩缩事件，则所需任务计数可能会设置为意外值。这是因为弹性伸缩将正在运行的任务计数视为当前容量，这是所需任务计数计算中使用的适当任务数的两倍。
- 使用 Fargate 启动类型或 `CODE_DEPLOY` 部署控制器类型的任务不支持 `DAEMON` 调度策略。
- 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。

所需的 IAM 权限

通过将 Amazon ECS 和 CodeDeploy API 组合使用以进行蓝绿部署。用户必须拥有这些服务的相应权限，才能在 AWS Management Console 或 AWS CLI 或 SDK 中使用 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 policy。有关更多信息，请参阅 AmazonECS_FullAccess。

```
{
    "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，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。

创建 Amazon ECS 服务示例 和 更新 Amazon ECS 服务示例 IAM policy 示例说明用户在 AWS Management Console 上使用 Amazon ECS 蓝/绿部署所需的权限。

使用蓝/绿部署方法部署 Amazon ECS 服务

了解如何创建一个 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。

> **Note**
>
> 为 AWS CloudFormation 增加了对执行蓝/绿部署的支持。有关更多信息，请参阅 AWS CloudFormation 用户指南中的使用 AWS CloudFormation 通过 CodeDeploy 进行Amazon ECS 蓝/绿部署。

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。
- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 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 the section called “创建 Virtual Private Cloud”。
- 创建 Amazon ECS CodeDeploy IAM 角色。有关更多信息，请参阅 Amazon ECS CodeDeploy IAM 角色。

步骤 1：创建 Application Load Balancer

使用蓝/绿部署类型的 Amazon ECS 服务需要使用应用 Application Load Balancer 或 Network Load Balancer。本教程使用一个 Application Load Balancer。

要创建 Application Load Balancer

1. 使用 create-load-balancer 命令创建 Application Load Balancer。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 Resource Name (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-cluster` 的集群。

```
aws ecs create-cluster \
    --cluster-name tutorial-bluegreen-cluster \
    --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-cluster
```

步骤 3：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。

```
{
    "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: #00FFFF;} </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-cluster",
    "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
```

使用以下命令获取负载均衡器的 DNS 名称。

```
aws elbv2 describe-load-balancers --name bluegreen-alb  --query
'LoadBalancers[*].DNSName'
```

在 Web 浏览器中，输入 DNS 名称，您应该可以看到以蓝色背景显示示例应用程序的网页。

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 Application Load Balancer 目标组以及 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-app \
        --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 Application Load Balancer 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。对于 `serviceRoleArn`，指定您的 Amazon ECS CodeDeploy IAM 角色的 ARN。有关更多信息，请参阅 Amazon ECS CodeDeploy IAM 角色。

```
{
   "applicationName": "tutorial-bluegreen-app",
   "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-cluster"
       }
   ]
}
```

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 6：创建和监控 CodeDeploy 部署

在创建 CodeDeploy 部署之前，请按如下方式更新 `fargate-task.json` 中的任务定义 `command`，以将示例应用程序的背景颜色更改为绿色。

```
{
    ...
        "containerDefinitions": [
            {
                ...
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #097969;} </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\""
                ]
            }
        ],
       ...
}
```

使用以下命令注册更新的任务定义。

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

现在，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。

要创建和监控 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。此示例使用更新的任务定义。

   ```
   version: 0.0
   Resources:
     - TargetService:
         Type: AWS::ECS::Service
         Properties:
           TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-
   definition/tutorial-task-def:2"
           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-bucket
   ```

   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-bucket/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-app",
    "deploymentGroupName": "tutorial-bluegreen-dg",
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bucket": "tutorial-bluegreen-bucket",
            "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"
}
```

3. 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-cluster:service-bluegreen \
--region us-east-1
```

   最初，部署状态为 `InProgress`。流量将定向到原始任务集，该任务集的 `taskSetLabel` 为 `BLUE`，状态为 `PRIMARY`，且 `trafficWeight` 为 `100.0`。替换任务集的 `taskSetLabel` 为 `GREEN`，状态为 `ACTIVE`，且 `trafficWeight` 为 `0.0`。您输入 DNS 名称所在的 Web 浏览器仍会以蓝色背景显示示例应用程序。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "status": "InProgress"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "status": "Pending"
        }
    ],
    "status": "InProgress",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
```

```
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        },
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        }
    ]
}
}
}
```

继续使用命令检索部署详细信息，直到部署状态为 Succeeded，如以下输出所示。流量现在被重定向到替换任务集，该任务集现在的状态为 PRIMARY，且 trafficWeight 为 100.0。刷新您在其中输入负载均衡器 DNS 名称的 Web 浏览器，现在您应该会看到绿色背景的示例应用程序。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
```

```
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "endTime": "2023-08-10T12:08:25.939000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "startTime": "2023-08-10T12:08:26.089000-05:00",
            "endTime":  "2023-08-10T12:08:26.403000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "startTime": "2023-08-10T12:08:26.926000-05:00",
            "endTime":  "2023-08-10T12:08:27.256000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "startTime": "2023-08-10T12:08:27.416000-05:00",
            "endTime": "2023-08-10T12:08:28.195000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "startTime": "2023-08-10T12:08:28.715000-05:00",
            "endTime":  "2023-08-10T12:08:28.994000-05:00",
            "status": "Succeeded"
        }
    ],
    "status": "Succeeded",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
```

```
            "taskSetLabel": "Blue"
        },
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        }
    ]
}
}
}
```

步骤 7：清除

完成本教程后，请清除与本教程关联的资源，以避免对您未使用的资源产生费用。

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-app \
     --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-app \
     --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 群集。

```
aws ecs delete-cluster \
    --cluster tutorial-bluegreen-cluster \
    --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-bucket/appspec.yaml
```

6. 使用 s3 rb 命令删除 Amazon S3 存储桶。

```
aws s3 rb s3://tutorial-bluegreen-bucket
```

7. 使用以下 delete-load-balancer 命令删除 Application Load Balancer。

```
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 Application Load Balancer 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
```

使用第三方控制器部署 Amazon ECS 服务

外部部署类型允许您使用任何第三方部署控制器，以完全控制 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 `DeleteTaskSet`）管理。每个 API 操作都会管理服务定义参数的一个子集。

`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。

外部部署注意事项

在使用外部部署类型时，请考虑以下内容：

- 支持的负载均衡器类型是应用程序负载均衡器或网络负载均衡器。
- Fargate 启动类型或 `EXTERNAL` 部署控制器类型均不支持 `DAEMON` 计划策略。

外部部署工作流

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`

   类型：字符串

   必需：是

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。一个集群中的服务名称必须唯一，但是您可以为一个区域或多个区域中多个集群中的服务提供相似的名称。

`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。

`tags`

类型： 对象数组

必需：否

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。该服务最多可应用 50 个标签。有关更多信息，请参阅 为 Amazon ECS 资源添加标签。

`key`

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

必需：否

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`

类型：字符串

长度约束：最小长度为 0。长度上限为 256。

必需：否

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使用 Amazon ECS 托管标签。有关更多信息，请参阅 使用标签记账。

`propagateTags`

类型：字符串

有效值：`TASK_DEFINITION` | `SERVICE`

必需：否

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。要在创建服务或任务以后将标签添加到任务，请使用 `TagResource` API 操作。

`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。

`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创建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": ""
        }
    ],
    "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`

   类型：字符串

   有效值：EC2 |FARGATE |EXTERNAL

   必需：否

   运行您的服务的启动类型。如果没有指定启动类型，将默认使用默认的 `capacityProviderStrategy`。有关更多信息，请参阅 Amazon ECS 启动类型。

   如果指定 `launchType`，必须省略 `capacityProviderStrategy` 参数。

`platformVersion`

类型：字符串

必需：否

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 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 适用于 Amazon ECS 的 Fargate Linux 平台版本。

> **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。使用外部部署控制器时，仅支持 Application Load Balancer 和 Network Load Balancer。如果您使用 Application Load Balancer，每个任务集仅允许一个 Application Load Balancer 目标组。

以下代码段显示要使用的示例 `loadBalancer` 对象。

```
"loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
]
```

> **Note**
>
> 指定 `loadBalancer` 对象时，必须指定 `targetGroupArn` 并忽略 `loadBalancerName` 参数。

`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 Fargate 启动类型联网的更多信息，请参阅Fargate 启动类型的 Amazon ECS 任务联网选项。

`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 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。

`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。

以下是为外部部署控制器创建任务集的 JSON 示例。

```
{
    "service": "",
    "cluster": "",
    "externalId": "",
    "taskDefinition": "",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
```

```
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "scale": {
        "value": null,
        "unit": "PERCENT"
    },
    "clientToken": ""
}
```

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。

使用负载均衡分配 Amazon ECS 服务流量

您可以选择将服务配置为使用弹性负载均衡来为服务中的任务平均分配流量。

> **Note**
>
> 当您使用任务集时，该集中的所有任务都必须配置为使用 Elastic Load Balancing 或不使用 Elastic Load Balancing。

托管在 AWS Fargate 上的 Amazon ECS 服务支持应用程序负载均衡器、网络负载均衡器和网关负载均衡器。请使用下表了解要使用的负载均衡器类型。

负载均衡器类型	在这些情况下使用
应用程序负载均衡器	路由 HTTP/HTTPS（或第 7 层）流量。 Application Load Balancers 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 Fargate 和 EC2 实例上托管的任务的支持。 • Application Load Balancers 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）。 • Application Load Balancer 支持基于路径的路由和优先级规则（以便多个服务可以在单个 Application Load Balancer 上使用相同的侦听器端口）。
网络负载均衡器	路由 TCP 或 UDP（或第 4 层）流量。
网关负载均衡器	路由 TCP 或 UDP（或第 4 层）流量。

负载均衡器类型	在这些情况下使用	
	使用虚拟设备，例如防火墙、入侵检测和防御系统以及深度数据包检测系统等。	

我们建议您为您的 Amazon ECS 服务使用应用程序负载均衡器，以便您可以利用这些最新功能，除非您的服务需要仅适用于网络负载均衡器或网关负载均衡器的功能。有关 Elastic Load Balancing 和这些类型的负载均衡器之间区别的更多信息，请参阅 Elastic Load Balancing 用户指南。

利用负载均衡器，您可以按实际用量付费。有关更多信息，请参阅 Elastic Load Balancing 定价。

优化 Amazon ECS 的负载均衡器运行状况检查参数

负载均衡器仅将请求路由至负载均衡器的可用区中的正常运行目标。每个目标都会注册到一个目标组。负载均衡器使用目标组的运行状况检查设置来检查每个目标的运行状况。在注册目标后，目标必须通过一次运行状况检查才会被视为正常。Amazon ECS 会监控负载均衡器。负载均衡器定期向 Amazon ECS 容器发送运行状况检查。Amazon ECS 代理会监控并等待负载均衡器报告容器的运行状况。它会在认为容器处于正常运行状态之前执行此操作。

两个 Elastic Load Balancing 运行状况检查参数对部署速度产生影响：

- 运行状况检查间隔：确定单个容器的运行状况检查之间的大致间隔时间（以秒为单位）。默认情况下，负载均衡器每 30 秒检查一次。

  此参数的名称为：

  - 在 Elastic Load Balancing API 中为 `HealthCheckIntervalSeconds`
  - 在 Amazon EC2 控制台上为间隔
- 运行状况正常阈值计数：确定将不正常容器运行状况视为正常之前需要的连续运行状况检查成功次数。默认情况下，负载均衡器需要五次通过运行状况检查才能报告目标容器运行状况正常。

  此参数的名称为：

  - 在 Elastic Load Balancing API 中为 `HealthyThresholdCount`
  - Amazon EC2 控制台上的运行状况正常阈值

在默认设置下，确定容器运行状况的总时间为 2 分钟 30 秒（`30 seconds * 5 = 150 seconds`）。

如果您的服务在不到 10 秒内启动并稳定下来，则可以加快运行状况检查过程。要加快此过程，请减少运行状况检查的次数和两次检查之间的间隔。

- `HealthCheckIntervalSeconds`（Elastic Load Balancing API 名称）或间隔（Amazon EC2 控制台名称）：5
- `HealthyThresholdCount`（Elastic Load Balancing API 名称）或运行状况正常阈值（Amazon EC2 控制台名称）：2

使用此设置，运行状况检查过程需要 10 秒，而默认时间为 2 分 30 秒。

有关 Elastic Load Balancing 运行状况检查参数的更多信息，请参阅《Elastic Load Balancing 用户指南》 中的目标组的运行状况检查。

优化 Amazon ECS 的负载均衡器连接耗尽参数

为了便于优化，客户端会保持与容器服务的活动连接。这样，来自该客户端的后续请求就可以重复使用现有连接。当您想停止流向容器的流量时，可以通知负载均衡器。负载均衡器会定期检查客户端是否关闭了保持活动连接。Amazon ECS 代理监控负载均衡器，并等待负载均衡器报告保持活动连接已关闭的情况（目标处于 `UNUSED` 状态）。

负载均衡器等待将目标移至 `UNUSED` 状态的时间为取消注册延迟。您可以配置以下负载均衡器参数来加快部署速度。

- `deregistration_delay.timeout_seconds`：300（默认值）

当您的服务响应时间小于一秒时，请将参数设置为以下值，使负载均衡器在中断客户端和后端服务之间的连接之前只等待 5 秒：

- `deregistration_delay.timeout_seconds`：5

> **Note**
>
> 当您的服务具有长期请求（例如文件上传缓慢或流式传输连接）时，请勿将该值设置为 5 秒。

SIGTERM 响应能力

Amazon ECS 首先向任务发送 SIGTERM 信号通知应用程序需要完成并关闭。然后，Amazon ECS 会发送一条 SIGKILL 消息。当应用程序忽略 SIGTERM 时，Amazon ECS 服务必须等待发送终止该进程的 SIGKILL 信号。

Amazon ECS 等待发送 SIGKILL 消息的时间由以下 Amazon ECS 代理选项确定：

- `ECS_CONTAINER_STOP_TIMEOUT`：30（默认值）

  有关容器代理参数的更多信息，请参阅 GitHub 上的 Amazon ECS 容器代理。

要加快等待时间，请将 Amazon ECS 代理参数设置为以下值：

> **Note**
>
> 如果您的应用程序花费的时间超过 1 秒，则请将该值乘以 2，然后将该数字用作值。

- `ECS_CONTAINER_STOP_TIMEOUT`：2

在本例中，Amazon ECS 会等待 2 秒钟让容器关闭，然后 Amazon ECS 会在应用程序未停止时发送 SIGKILL 消息。

您也可以修改应用程序代码以捕获 SIGTERM 信号，并对其做出反应。以下是 JavaScript 中的示例：

```
process.on('SIGTERM', function() {
  server.close();
})
```

此代码会导致 HTTP 服务器停止侦听任何新请求，完成回答任何正在处理的请求，然后 Node.js 进程终止。这是因为它的事件循环无事可做。鉴于此，如果该进程只需 500 毫秒即可完成其正在处理的请求，则它会提前终止，而不必等待停止超时并收到 SIGKILL。

使用 Amazon ECS 的应用程序负载均衡器

Application Load Balancer 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您的群集中每个容器实例上的一个或多个端口。Application Load Balancer 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，在最新的经 Amazon ECS 优化的 AMI 上，为 32768 到 61000）中动态选择主机端口。在任务负载均衡器时，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，请参阅 Application Load Balancer 用户指南。

有关设置参数以加快部署速度的最佳实践的信息，请参阅：

- 优化 Amazon ECS 的负载均衡器运行状况检查参数
- 优化 Amazon ECS 的负载均衡器连接耗尽参数

将应用程序负载均衡器与 Amazon ECS 结合使用时，请考虑以下事项：

- Amazon ECS 需要与服务相关的 IAM 角色，该角色提供在创建和停止任务时向负载均衡器注册和注销目标所需的权限。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。
- 目标组必须将 IP 地址类型设置为 IPv4。
- 如果服务具有使用 `awsvpc` 网络模式的任务，在为服务创建目标组时，必须选择 `ip` 作为目标类型，而不是`instance`。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。
- 如果服务需要访问多个负载均衡端口（如某项 HTTP/HTTPS 服务需要端口 80 和端口 443），可以配置两个侦听器。一个侦听器负责将请求转发给服务的 HTTPS，另一个侦听器负责将 HTTP 请求重定向到适当的 HTTPS 端口。有关更多信息，请参阅 Application Load Balancers 用户指南中的创建 Application Load Balancer 的侦听器。
- 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。
- 创建服务后，无法从 AWS Management Console 更改负载均衡器配置。您只能使用 AWS Copilot、AWS CloudFormation、AWS CLI 或 SDK 来修改 ECS 滚动部署控制器的负载均衡器配置，而不是 AWS CodeDeploy 蓝/绿或外部控制器。当您添加、更新或删除负载均衡器配置时，Amazon ECS 会使用更新后的 Elastic Load Balancing 配置启动新部署。这将导致任务注册到负载均衡器或从负载均衡器取消注册。我们建议您在更新 Elastic Load Balancing 配置之前在测试环境中对此进行验证。有关如何修改配置的信息，请参阅 Amazon Elastic Container Service API 参考中的 UpdateService。
- 如果服务任务未达到负载均衡器运行状况检查标准，则系统会停止并重启该任务。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 对 Amazon ECS 中的服务负载均衡器进行故障排除。

- 您的任务和负载均衡器必须都在相同 VPC 中。
- 为每项服务使用唯一的目标组。

  为多个服务使用同一个目标组可能会导致服务部署期间出现问题。

有关如何创建应用程序负载均衡器的信息，请参阅应用程序负载均衡器中的创建应用程序负载均衡器

对 Amazon ECS 使用网络负载均衡器

Network Load Balancer 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。它将不修改标头的情况下转发请求。Network Load Balancers 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，在最新的经 Amazon ECS 优化的 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 Network Load Balancer，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 Network Load Balancer 的更多信息，请参阅 Network Load Balancers 用户指南。

有关设置参数以加快部署速度的最佳实践的信息，请参阅：

- 优化 Amazon ECS 的负载均衡器运行状况检查参数
- 优化 Amazon ECS 的负载均衡器连接耗尽参数

将网络负载均衡器与 Amazon ECS 结合使用时，请考虑以下事项：

- Amazon ECS 需要与服务相关的 IAM 角色，该角色提供在创建和停止任务时向负载均衡器注册和注销目标所需的权限。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。
- 附加到一项服务的目标组不能超过五个。
- 如果服务具有使用 `awsvpc` 网络模式的任务，在为服务创建目标组时，必须选择 `ip` 作为目标类型，而不是`instance`。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。
- 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。
- 创建服务后，无法从 AWS Management Console 更改负载均衡器配置。您只能使用 AWS Copilot、AWS CloudFormation、AWS CLI 或 SDK 来修改 ECS 滚动部署控制器的负载均衡器配置，而不是 AWS CodeDeploy 蓝/绿或外部控制器。当您添加、更新或删除负载均衡器配置

时，Amazon ECS 会使用更新后的 Elastic Load Balancing 配置启动新部署。这将导致任务注册到负载均衡器或从负载均衡器取消注册。我们建议您在更新 Elastic Load Balancing 配置之前在测试环境中对此进行验证。有关如何修改配置的信息，请参阅 Amazon Elastic Container Service API 参考中的 UpdateService。

- 如果服务任务未达到负载均衡器运行状况检查标准，则系统会停止并重启该任务。此过程将持续到您的服务达到预期的运行任务的数量。
- 当使用将 IP 地址配置为目标且禁用的客户端 IP 保留的网络负载均衡器时，请求将被视为来自网关负载均衡器的私有 IP 地址。这意味着，只要您允许目标安全组中的传入请求和运行状况检查，网关负载均衡器后面的服务就会有效地向世界开放。
- 对于 Fargate 任务，您必须使用平台版本 `1.4.0`（Linux）或 `1.0.0`（Windows）。
- 如果您的支持负载均衡器的服务出现问题，请参阅 对 Amazon ECS 中的服务负载均衡器进行故障排除。
- 您的任务和负载均衡器必须都在相同 VPC 中。
- 网络负载均衡器客户端 IP 地址保留与 Fargate 目标兼容。
- 为每项服务使用唯一的目标组。

  为多个服务使用同一个目标组可能会导致服务部署期间出现问题。

有关如何创建网络负载均衡器的信息，请参阅网络负载均衡器中的创建网络负载均衡器

> **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。

对 Amazon ECS 使用网关负载均衡器

Gateway Load Balancer 在 Open Systems Interconnection (OSI) 模型的第四层运行，网络层。它监听所有端口上的所有 IP 数据包，并将流量转发到监听程序规则中指定的目标组。使用 5 元组（对于 TCP/UDP 流）或 3 元组（对于非 TCP/UDP 流）来保持流向特定目标设备的粘性。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，在最新的经 Amazon ECS 优化的 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关负载均衡器，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，请参阅网关负载均衡器中的什么是网关负载均衡器。

有关设置参数以加快部署速度的最佳实践的信息，请参阅：

- 优化 Amazon ECS 的负载均衡器运行状况检查参数
- 优化 Amazon ECS 的负载均衡器连接耗尽参数

将网关负载均衡器与 Amazon ECS 结合使用时，请考虑以下事项：

- Amazon ECS 需要与服务相关的 IAM 角色，该角色提供在创建和停止任务时向负载均衡器注册和注销目标所需的权限。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。
- 如果服务具有使用 `awsvpc` 网络模式的任务，在为服务创建目标组时，必须选择 `ip` 作为目标类型，而不是`instance`。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。
- 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。
- 创建服务后，无法从 AWS Management Console 更改负载均衡器配置。您只能使用 AWS Copilot、AWS CloudFormation、AWS CLI 或 SDK 来修改 ECS 滚动部署控制器的负载均衡器配置，而不是 AWS CodeDeploy 蓝/绿或外部控制器。当您添加、更新或删除负载均衡器配置时，Amazon ECS 会使用更新后的 Elastic Load Balancing 配置启动新部署。这将导致任务注册到负载均衡器或从负载均衡器取消注册。我们建议您在更新 Elastic Load Balancing 配置之前在测试环境中对此进行验证。有关如何修改配置的信息，请参阅 Amazon Elastic Container Service API 参考中的 UpdateService。
- 如果服务任务未达到负载均衡器运行状况检查标准，则系统会停止并重启该任务。此过程将持续到您的服务达到预期的运行任务的数量。
- 当使用将 IP 地址配置为目标的网关负载均衡器时，请求将被视为来自网关负载均衡器的私有 IP 地址。这意味着，只要您允许目标安全组中的传入请求和运行状况检查，网关负载均衡器后面的服务就会有效地向世界开放。
- 对于 Fargate 任务，您必须使用平台版本 `1.4.0`（Linux）或 `1.0.0`（Windows）。
- 如果您的支持负载均衡器的服务出现问题，请参阅 对 Amazon ECS 中的服务负载均衡器进行故障排除。
- 您的任务和负载均衡器必须都在相同 VPC 中。

- 为每项服务使用唯一的目标组。

  为多个服务使用同一个目标组可能会导致服务部署期间出现问题。

有关如何创建网关负载均衡器的信息，请参阅网关负载均衡器中的创建网关负载均衡器

> **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。

将多个目标组注册到 Amazon ECS 服务

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。

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 Management Console查看服务以及注册到服务的目标组。您必须使用 UpdateService 修改现有服务的负载均衡器配置。

可以使用以下格式在服务定义中指定多个目标组。有关服务定义的完整语法，请参阅服务定义模板。

```
"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
   }
]
```

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：

- 对于使用 Application Load Balancer 或 Network Load Balancer 的服务，附加到一个服务的目标组不能超过五个。
- 仅在以下条件下，才支持在服务定义中指定多个目标组：
  - 服务必须使用 Application Load Balancer 或 Network Load Balancer。
  - 该服务必须使用滚动更新 (ECS) 部署控制器类型。
- 包含使用 Fargate 和 EC2 启动类型的任务的服务支持多个目标组。
- 当您创建一个指定多个目标组的服务时，必须创建 Amazon ECS 服务相关角色。通过省略 API 请求中的 `role` 参数或 AWS CloudFormation 中的 `Role` 属性来创建角色。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。

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。有关服务定义的完整语法，请参阅服务定义模板。

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，另一个用于面向互联网的流量。

```
"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
   }
]
```

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
   }
]
```

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
   }
]
```

自动扩展 Amazon ECS 服务

Automatic scaling 是指自动增加或减少 Amazon ECS 服务中所需任务数的功能。Amazon ECS 利用 Application Auto Scaling 服务来提供此功能。有关更多信息，请参阅 Application Auto Scaling 用户指南。

Amazon ECS 发布 CloudWatch 指标与服务的平均 CPU 和内存使用率。有关更多信息，请参阅 Amazon ECS 服务利用率指标。您可以使用这些指标和其他 CloudWatch 指标扩展您的服务（添加更多任务）以应对高峰期的高需求，并缩减您的服务（运行更少的任务）以降低低利用率期间的成本。

Amazon ECS Service Auto Scaling 支持以下类型的自动扩展：

- 使用目标指标值扩展您的 Amazon ECS 服务 – 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。
- 使用基于 CloudWatch 警报的预定义增量扩展您的 Amazon ECS 服务 – 根据一组扩缩调整，增加或减少服务运行的任务数，这些调整称为分布调整，将根据警报严重程度发生变化。
- 使用计划扩展 Amazon ECS 服务 – 根据日期和时间增加或减少服务运行的任务数。

注意事项

使用扩缩策略时，请注意以下事项：

- Amazon ECS 每隔 1 分钟向 CloudWatch 发送一次指标数据。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。
- 缩放策略支持冷却期。这是等待上一个扩展活动生效的秒数。
  - 对于扩展事件，目的是持续（但不过度）向外扩展。Service Auto Scaling 使用步进扩展策略成功向外扩展后，它将开始计算冷却时间。除非启动更大的横向扩展或冷却时间结束，否则扩缩策略不会再次增加所需容量。尽管此向外扩展冷却时间有效，但启动向外扩展活动所添加的容量将计算为下一个向外扩展活动所需容量的一部分。

  - 对于缩减事件，目的是以保守方式进行缩减以保护应用程序的可用性，因此在冷却时间过期之前阻止缩减活动。但是，如果另一个警报在横向缩减冷却时间内启动了横向扩展活动，Service Auto Scaling 将立即对目标进行横向扩展。在这种情况下，缩减冷却时间停止而不完成。
- 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缩策略和警报，Service 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，并且警报启动了横向扩展活动，则 Service 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横向扩展。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，并且警报启动了横向缩减活动，则 Service 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横向缩减。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计数是 Service Auto Scaling 用作其起点的值（而不是预期数量）。这是应有的处理容量。这可防止无法满足的过度（失控）扩展，例如，没有足够的容器实例资源来放置其他任务。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。
- 如果您希望在没有要完成的工作时将任务计数扩展到零，请将最小容量设置为 0。对于目标跟踪扩展策略，当实际容量为 0 且度量指示存在工作负载需求时，Service Auto Scaling 会等待发送一个数据点，然后再扩展。在这种情况下，它将按可能的最小量扩展作为起点，然后根据实际运行的任务计数恢复扩展。
- Application Auto Scaling 可在 Amazon ECS 部署正在进行时关闭横向缩减流程。但是，在部署过程中，除非暂停，否则将继续发生扩展进程。有关更多信息，请参阅 服务自动扩展和部署。
- Amazon ECS 任务有数个 Application Auto Scaling 选项。目标跟踪是最易于使用的模式。有了它，您所需要做的就是为指标设置一个目标值，例如 CPU 平均利用率。然后，自动定标器会自动管理实现该值所需的任务数量。使用分步扩展，您可以更快地对需求变化做出反应，因为您可以定义扩展指标的特定阈值，以及在超过阈值时要添加或删除的任务数量。并且，更重要的是，您可以通过最大限度地减少突破阈值警报的时间来对需求变化做出快速反应。

优化 Amazon ECS 服务自动扩缩

Amazon ECS 服务是一个托管任务集合。每项服务都有关联的任务定义、所需的任务计数和可选的放置策略。Amazon ECS 服务自动扩缩是通过 Application Auto Scaling 服务实现的。Application Auto Scaling 使用 CloudWatch 指标作为扩展指标的来源。它还使用 CloudWatch 警报来设置何时横向缩减或横向扩展服务的阈值。您可以通过设置指标目标（称为目标跟踪扩缩）或通过指定阈值（称为分步扩缩）来提供扩缩阈值。配置 Application Auto Scaling 后，它会持续计算服务的相应所需任务数。它还会在所需的任务计数发生变化时通过横向扩展或横向缩减来通知 Amazon ECS。

要有效地使用服务自动扩缩，必须选择适当的扩缩指标。

如果需求预计将大于当前容量，则应横向扩展应用程序。相反，当资源超过需求时，可以横向缩减应用程序以节省成本。

识别指标

要有效地进行扩展，标识一个表明利用率或饱和度的指标至关重要。此指标必须显示以下属性才能用于扩缩。

- 该指标必须与需求相关联。当资源保持稳定，但需求发生变化时，指标值也必须发生变化。当需求增加或减少时，该指标也应增加或减少。
- 指标值必须与容量成比例地横向缩减。当需求保持不变时，增加更多资源必须导致指标值按比例变化。因此，将任务数增加一倍应该会使该指标减少 50%。

标识利用率指标的最佳方法是在预生产环境（例如暂存环境）中进行负载测试。商业和开源负载测试解决方案广泛可用。这些解决方案通常可以生成合成负载，或模拟真实的用户流量。

要开始负载测试过程，请为应用程序的利用率指标构建控制面板。这些指标包括 CPU 利用率、内存利用率、I/O 操作、I/O 队列深度和网络吞吐量。您可以使用 Container Insights 等服务收集这些指标。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。在此过程中，请确保收集应用程序响应时间或工作完成率指标并绘制其图形。

从一个很小的请求或作业插入率开始。将此速率保持稳定几分钟，让您的应用程序进行预热。然后，慢慢提高速率并使其保持稳定几分钟。重复此周期，每次都提高速率，直到应用程序的响应或完成时间太慢而无法达到服务级目标（SLO）。

进行负载测试时，请检查每个利用率指标。随负载增加而增加的指标最适合作为最佳利用率指标。

接下来，识别已达到饱和度的资源。同时，还要检查利用率指标，看看哪个指标首先在高水平上趋于平缓，或者达到峰值，然后首先使应用程序崩溃。例如，如果随着负载增加，CPU 利用率从 0% 增加到 70-80%，然后在增加更多负载后仍保持在该水平，则可以肯定地说 CPU 已饱和。根据 CPU 架构的不同，它可能永远不会达到 100%。例如，假设内存利用率随负载的增加而增加，然后您的应用程序在达到任务或 Amazon EC2 实例内存限制时突然崩溃。这种情况下，很可能是内存已被完全消耗。您的应用程序可能会消耗多种资源。因此，请选择表示首先耗尽的资源的指标。

最后，在任务数或 Amazon EC2 实例数增加一倍后，再次尝试负载测试。假设关键指标的增加或减少速度是以前的一半。如果是这种情况，则指标与容量成正比。这是自动扩缩的一个很好的利用率指标。

现在考虑一下这个假设场景。假设您对应用程序进行负载测试，并发现每秒 100 个请求时 CPU 利用率最终达到 80%。当增加更多负载时，CPU 利用率不会再提高。但是，它确实会使您的应用程序的响应速度变慢。然后，您再次运行负载测试，将任务数增加一倍，但将速率保持在之前的峰值水平。如果您发现平均 CPU 利用率降至 40% 左右，则平均 CPU 利用率是扩缩指标的理想选择。另一方面，如果增加任务数后 CPU 利用率保持在 80%，则平均 CPU 利用率不是一个很好的扩缩指标。这种情况下，需要进行更多的研究才能找到合适的指标。

常见的应用程序模型和扩缩属性

各种软件都可以在 AWS 上运行。许多工作负载是自主开发的，而另一些工作负载则基于常见的开源软件。无论它们来自哪里，我们都观察到了一些常见的服务设计模式。如何有效扩展很大程度上取决于模式。

高效的 CPU 绑定服务器

除了 CPU 和网络吞吐量之外，高效的 CPU 绑定服务器几乎不利用任何资源。每个请求都可以由应用程序单独处理。请求不依赖于数据库等其他服务。应用程序可以处理成千上万个并发请求，并且可以高效地利用多个 CPU 来执行此操作。每个请求或由内存开销较低的专用线程提供服务，或者在为请求提供服务的每个 CPU 上运行一个异步事件循环。应用程序的每个副本都同样能够处理请求。可能在 CPU 之前耗尽的唯一资源是网络带宽。在 CPU 绑定服务中，即使在吞吐量峰值下，内存利用率也只是可用资源的一小部分。

这种类型的应用程序适用于基于 CPU 的自动扩缩。该应用程序在扩缩方面具有最大的灵活性。可以通过向其提供更大的 Amazon EC2 实例或 Fargate vCPU 来对其进行纵向扩展。而且，还可以通过添加更多副本对其进行水平扩展。添加更多副本或将实例大小增加一倍，可将平均 CPU 利用率（相对于容量）降低一半。

如果您为此应用程序使用 Amazon EC2 容量，则请考虑将其放置在计算优化型实例上，例如 c5 或 c6g 系列。

高效的内存绑定服务器

高效的内存绑定服务器为每个请求分配大量内存。在最大并发（但不一定是吞吐量）下，内存会在 CPU 资源耗尽之前耗尽。请求结束时，与请求关联的内存会被释放。只要有可用内存，就可以接受额外请求。

这种类型的应用程序适用于基于内存的自动扩缩。该应用程序在扩缩方面具有最大的灵活性。可以通过向其提供更大的 Amazon EC2 或 Fargate 内存资源来对其进行纵向扩展。而且，还可以通过添加更多副本对其进行水平扩展。添加更多副本或将实例大小增加一倍，可将平均内存利用率（相对于容量）降低一半。

如果您为此应用程序使用 Amazon EC2 容量，请考虑将其放置在内存优化型实例上，例如 r5 或 r6g 系列。

某些受内存限制的应用程序在某项请求结束时不会释放与该请求关联的内存，因此并发度的降低不会导致使用的内存减少。为此，不建议您使用基于内存的扩缩。

基于工作线程的服务器

基于工作线程的服务器一个接一个地处理每个工作线程的请求。工作线程可以是轻量级线程，例如 POSIX 线程。它们也可以是重量级线程，例如 UNIX 进程。无论它们是哪个线程，应用程序可以支持的最大并发度总是存在的。通常，并发限制根据可用的内存资源按比例设置。如果达到并发限制，则会将其他请求放入积压队列中。如果积压队列溢出，则其他传入请求会立即被拒绝。符合这种模式的常见应用程序包括 Apache Web 服务器和 Gunicorn。

请求并发度通常是扩缩此应用程序的最佳指标。由于每个副本都有并发限制，因此务必在达到平均限制之前进行横向扩展。

获取请求并发指标的最佳方法是让您的应用程序将其报告给 CloudWatch。应用程序的每个副本都可以将并发请求数量作为自定义指标高频率发布。建议将频率设置为至少每分钟一次。收集多个报告后，您可以使用平均并发度作为扩缩指标。此指标通过将总并发度除以副本数计算得出。例如，如果总并发度为 1000，副本数为 10，则平均并发度为 100。

如果您的应用程序位于应用程序负载均衡器之后，则您也可以使用负载均衡器的 ActiveConnectionCount 指标作为扩缩指标中的一个因子。必须将 ActiveConnectionCount 指标除以副本数才能得出平均值。必须使用平均值进行扩缩，而不是使用原始计数值。

为了使该设计发挥最佳效果，在低请求速率下，响应延迟的标准差应该较小。建议在需求低迷时期，在短时间内答复大多数请求，并且没有多少请求需要的时间比平均响应时间长得多。平均响应时间应接近第 95 个百分位响应时间。否则，可能会导致发生队列溢出。这会导致错误。建议您在必要时提供其他副本，以缓解溢出风险。

等待服务器

等待服务器会对每个请求进行一些处理，但它高度依赖于一个或多个下游服务运行。容器应用程序经常会大量使用下游服务，例如数据库和其他 API 服务。这些服务可能需要一些时间才能做出响应，尤其是在高容量或高并发场景中。这是因为这些应用程序往往使用很少的 CPU 资源，并利用可用内存方面的最大并发性。

等待服务既适用于内存绑定的服务器模式，也适用于基于工作线程的服务器模式，具体取决于应用程序的设计方式。如果应用程序的并发度仅受内存限制，则应使用平均内存利用率作为扩缩指标。如果应用程序的并发度基于工作线程限制，则应使用平均并发度作为扩缩指标。

基于 Java 的服务器

如果您的基于 Java 的服务器受 CPU 约束，并且与 CPU 资源成比例地扩展，则它可能适合高效的 CPU 绑定服务器模式。如果是这种情况，则平均 CPU 利用率可能适合作为扩缩指标。但是，许多 Java 应用程序不受 CPU 限制，因此难以扩展。

为了获得最佳性能，建议您为 Java 虚拟机（JVM）堆分配尽可能多的内存。最新版本的 JVM（包括 Java 8 更新 191 或更高版本）会自动将堆大小设置得尽可能大，以容纳到容器中。这意味着，在 Java 中，内存利用率很少与应用程序利用率成正比。随着请求速率和并发度提高，内存利用率保持不变。因此，不建议根据内存利用率扩缩基于 Java 的服务器。相反，通常建议按 CPU 利用率扩缩。

某些情况下，基于 Java 的服务器在耗尽 CPU 之前会遇到堆耗尽的情况。如果您的应用程序在高并发时容易出现堆耗尽的情况，则平均连接数是最好的扩缩指标。如果您的应用程序在高吞吐量时容易出现堆耗尽的情况，则平均请求率是最好的扩缩指标。

使用其他垃圾回收运行时的服务器

许多服务器应用程序都基于执行垃圾回收的运行时，例如 .NET 和 Ruby。这些服务器应用程序可能符合前面描述的模式之一。但是，与 Java 一样，不建议根据内存扩缩这些应用程序，因为它们观察到的平均内存利用率通常与吞吐量或并发度不相关。

对于这些应用程序，如果应用程序受 CPU 限制，建议您按 CPU 利用率扩展。否则，建议您根据负载测试结果按平均吞吐量或平均并发度扩展。

作业处理器

许多工作负载都涉及异步作业处理。它们包括不实时接收请求，而是通过订阅工作队列来接收作业的应用程序。对于这些类型的应用程序，正确的扩缩指标几乎总是队列深度。队列增长表明待处理的工作超过了处理能力，而空队列则表示容量大于要做的工作。

诸如 Amazon SQS 和 Amazon Kinesis Data Streams 之类的 AWS 消息收发服务提供了可用于扩缩的 CloudWatch 指标。对于 Amazon SQS，`ApproximateNumberOfMessagesVisible` 是最好的指标。对于 Kinesis Data Streams，可以考虑使用 `MillisBehindLatest` Kinesis Client Library（KCL）发布的指标。在使用该指标进行扩缩之前，应在所有使用者之间求它的平均值。

服务自动扩展和部署

Application Auto Scaling 可在 Amazon ECS 部署正在进行时关闭横向缩减流程。但是，在部署过程中，除非暂停，否则将继续发生扩展进程。如果要在部署正在进行的过程中暂停向外扩展进程，请执行以下步骤。

1. 调用 describe-scalable-targets 命令，在 Application Auto Scaling 中指定与可扩展目标关联的服务的资源 ID（例如：`service/default/sample-webapp`）。记录输出。调用下一个命令时，您将用到它。
2. 调用 register-scalable-target 命令，指定资源 ID、命名空间和可伸缩维度。指定 `true` 代表 `DynamicScalingInSuspended` 和 `DynamicScalingOutSuspended`。
3. 部署完成后，您可以调用 register-scalable-target 命令恢复扩展。

有关更多信息，请参阅暂停和恢复 Application Auto Scaling 的扩缩。

使用目标指标值扩展您的 Amazon ECS 服务

在使用目标跟踪扩展策略时，您可以选择一个指标并设置一个目标值。Amazon ECS Service Auto Scaling 创建和管理控扩缩策略的 CloudWatch 警报，并根据指标和目标值计算扩缩调整。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。

注意事项

使用目标跟踪策略时，请注意以下事项：

- 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 Service 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。
- 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。
- Application Auto Scaling 可在 Amazon ECS 部署正在进行时关闭横向缩减流程。但是，在部署过程中，除非暂停，否则将继续发生扩展进程。有关更多信息，请参阅 服务自动扩展和部署。
- 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。Service Auto Scaling 的目的是始终优先考虑可用性，因此其行为会有所不同，具体取决于目标跟踪策略是否已准备好扩展或缩减。如果任何目标跟踪策略已准备好进行横向扩展，它将横向扩展服务；但仅在所有目标跟踪策略（启用了横向缩减部分）准备好横向缩减时才执行横向缩减。
- 请勿编辑或删除 Service Auto Scaling 为目标跟踪缩放策略管理的 CloudWatch 报警。当您删除扩展策略时，Service Auto Scaling 将自动删除相应的警报。

- 蓝绿部署类型不支持目标跟踪扩展策略的 `ALBRequestCountPerTarget` 指标。

有关目标跟踪扩展策略的更多信息，请参阅《Application Auto Scaling 用户指南》中的目标跟踪扩展策略。

要使用 Amazon ECS 控制台为 Amazon ECS 服务配置目标扩缩策略

1. 除了用于创建和更新服务的标准 IAM 权限之外，您还需要额外权限。有关更多信息，请参阅 Amazon ECS 服务自动扩缩所需的 IAM 权限。
2. 您可以在创建或更新服务时配置扩缩策略。有关更多信息，请参阅以下章节之一：
   - 使用定义的参数创建服务— 创建新服务
   - 使用控制台更新 Amazon ECS 服务— 更新现有服务

要使用 AWS CLI 为 Amazon ECS 服务配置目标扩缩策略

1. 除了用于创建和更新服务的标准 IAM 权限之外，您还需要额外权限。有关更多信息，请参阅 Amazon ECS 服务自动扩缩所需的 IAM 权限。
2. 使用 register-scalable-target 命令将 Amazon ECS 服务注册为可扩展目标。
3. 使用 put-scaling-policy 命令创建扩展策略。

使用基于 CloudWatch 警报的预定义增量扩展您的 Amazon ECS 服务

在使用步进扩缩策略时，您可以指定 CloudWatch 警报以启动扩缩过程。例如，如果您希望在 CPU 利用率达到特定水平时横向扩展，可以使用提供的 `CPUUtilization` 指标创建警报。在创建步进扩展策略时，您必须指定以下扩展调整类型之一：

- 增加 – 按指定的容量单位数量或当前容量的指定百分比来增加任务数量。
- 移除 – 按指定的容量单位数量或当前容量的指定百分比来减少任务数量。
- 设定为 - 将任务数量设为指定的容量单位数量。

例如，假设目标容量和执行容量为 10，扩展策略加 1。当突破警报阈值时，自动扩缩过程为 10 增加 1 得到 11，因此 Amazon ECS 为该服务启动 1 个任务。

强烈建议您使用目标跟踪扩缩策略，根据类似于平均 CPU 利用率或每个目标的平均请求数等指标进行扩展。使用目标跟踪，可以通过在容量增加时减少以及在容量减少时增加的指标，按比例横向扩展或缩减任务数。这有助于确保 Service Auto Scaling 密切遵循应用程序的需求曲线。

有关步进扩缩策略及其工作原理的概述，请参阅《Application Auto Scaling 用户指南》中的步进扩缩策略。阅读本简介后，请参阅以下部分，了解如何使用控制台和 AWS Command Line Interface 为 Amazon ECS 配置分步扩缩。

要使用 Amazon ECS 控制台为 Amazon ECS 服务配置步进扩缩策略

1. 除了用于创建和更新服务的标准 IAM 权限之外，您还需要额外权限。有关更多信息，请参阅 Amazon ECS 服务自动扩缩所需的 IAM 权限。
2. 您可以在创建或更新服务时配置扩缩策略。有关更多信息，请参阅以下章节之一：
   - 使用定义的参数创建服务— 创建新服务
   - 使用控制台更新 Amazon ECS 服务— 更新现有服务

要使用 AWS CLI 为 Amazon ECS 服务配置步进扩展策略

1. 除了用于创建和更新服务的标准 IAM 权限之外，您还需要额外权限。有关更多信息，请参阅 Amazon ECS 服务自动扩缩所需的 IAM 权限。
2. 使用 register-scalable-target 命令将 Amazon ECS 服务注册为可扩展目标。
3. 使用 put-scaling-policy 命令创建扩展策略。
4. 使用 put-metric-alarm 命令创建启动扩缩策略的警报。

使用计划扩展 Amazon ECS 服务

通过计划扩缩，您可以根据可预测的负载变化，通过创建在特定时间增加或减少容量的计划操作，为应用程序设置自动扩缩。这使您可以主动扩缩应用程序，以适应可预测的负载变化。

利用这些计划的扩缩操作，您可以优化成本和性能。您的应用程序有足够的容量来处理一周中间的流量高峰，但不会在其他时间过度配置不需要的容量。

您可以同时使用计划的扩缩和扩缩策略，以获得主动和被动扩缩方法的优势。运行计划的扩缩操作后，扩缩策略可以继续决定是否进一步扩缩容量。这有助于确保您有足够的容量来处理应用程序的负载。当您的应用程序扩展以满足需求时，当前容量必须在计划操作设置的最小容量和最大容量范围内。

您可以使用 AWS CLI 配置计划扩缩。有关使用计划扩缩的更多信息，请参阅《Application Auto Scaling 用户指南》中的计划扩缩。

互连 Amazon ECS 服务

在 Amazon ECS 任务中运行的应用程序通常需要接收来自互联网的连接或连接到在 Amazon ECS 服务中运行的其他应用程序。如果您需要来自互联网的外部连接，我们建议您使用 Elastic Load Balancing。有关集成负载均衡的更多信息，请参阅 the section called “使用负载均衡分配服务流量”。

如果您需要一个应用程序来连接到在 Amazon ECS 服务中运行的其他应用程序，Amazon ECS 会提供以下无需负载均衡器的方法来实现此目的：

- Amazon ECS Service Connect

  建议使用 Service Connect，它为服务发现、连接和流量监控提供 Amazon ECS 配置。借助 Service Connect，您的应用程序可以使用短名称和标准端口连接到同一集群中的 Amazon ECS 服务，也可以连接到其他集群（包括同一 AWS 区域 的不同 VPC 中）中的 Amazon ECS 服务。

  当您使用 Service Connect 时，Amazon ECS 会管理服务发现的所有部分：创建可发现的名称，在任务开始和停止时动态管理每个任务的条目，在配置为发现名称的每个任务中运行代理。您的应用程序可以通过使用 DNS 名称的标准功能，并建立连接来查找这些名称。如果您的应用程序已执行此操作，则您无需修改应用程序即可使用 Service Connect。

  您在每项服务和任务定义中都提供完整的配置。Amazon ECS 在每个服务部署中管理对该配置的更改，以确保部署中的所有任务以相同的方式运行。例如，DNS 作为服务发现的一个常见问题是控制迁移。如果您将 DNS 名称更改为指向新的替换 IP 地址，则在所有客户端开始使用新服务之前，可能需要最大的 TTL 时间。使用 Service Connect，客户端部署将通过替换客户端任务来更新配置。您可以像配置任何其他部署一样配置部署断路器和其他部署配置，以对 Service Connect 更改造成影响。

  有关更多信息，请参阅 使用 Service Connect 连接具有短名称的 Amazon ECS 服务。

- Amazon ECS 服务发现

  服务间通信的另一种方法是使用服务发现进行直接通信。在这种方法中，您可以使用与 Amazon ECS 的 AWS Cloud Map 服务发现集成。使用服务发现，Amazon ECS 将已启动的任务列表同步到 AWS Cloud Map，后者将维护一个 DNS 主机名，该主机名解析为来自该特定服务的一个或多个任务的内部 IP 地址。Amazon VPC 中的其他服务可以使用此 DNS 主机名，以通过其内部 IP 地址将流量直接发送到另一个容器。

这种服务间的通信方法可实现低延迟。容器之间没有额外的组件。流量直接从一个容器传递到另一个容器。

这种方法适用于使用 awsvpc 网络模式时，在这种模式下，每项任务都有自己唯一的 IP 地址。大多数软件仅支持使用 DNS A 记录，这些记录可直接解析为 IP 地址。使用 awsvpc 网络模式时，每项任务的 IP 地址都是一个 A 记录。但是，如果您使用 bridge 网络模式，则多个容器可能会共享相同的 IP 地址。此外，动态端口映射会导致容器在该单个 IP 地址上被随机分配端口号。此时，A 记录已不足以进行服务发现。您还必须使用 SRV 记录。这种类型的记录可以同时跟踪 IP 地址和端口号，但需要您相应地配置应用程序。您使用的某些预建应用程序可能不支持 SRV 记录。

awsvpc 网络模式的另一个优点是，您的每项服务都有一个唯一的安全组。您可以将此安全组配置为只允许来自需要与该服务通信的特定上游服务的传入连接。

使用服务发现进行服务间直接通信的主要缺点是，您必须实现额外的逻辑才能进行重试和处理连接失败。DNS 记录具有生存时间（TTL），可控制其缓存时长。更新 DNS 记录和缓存过期需要一些时间，这样您的应用程序可以获取 DNS 记录的最新版本。因此，您的应用程序最终可能会将 DNS 记录解析为指向另一个不再存在的容器。您的应用程序需要处理重试，并具有忽略不良后端的逻辑。

有关更多信息，请参阅 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务

网络模式兼容性表

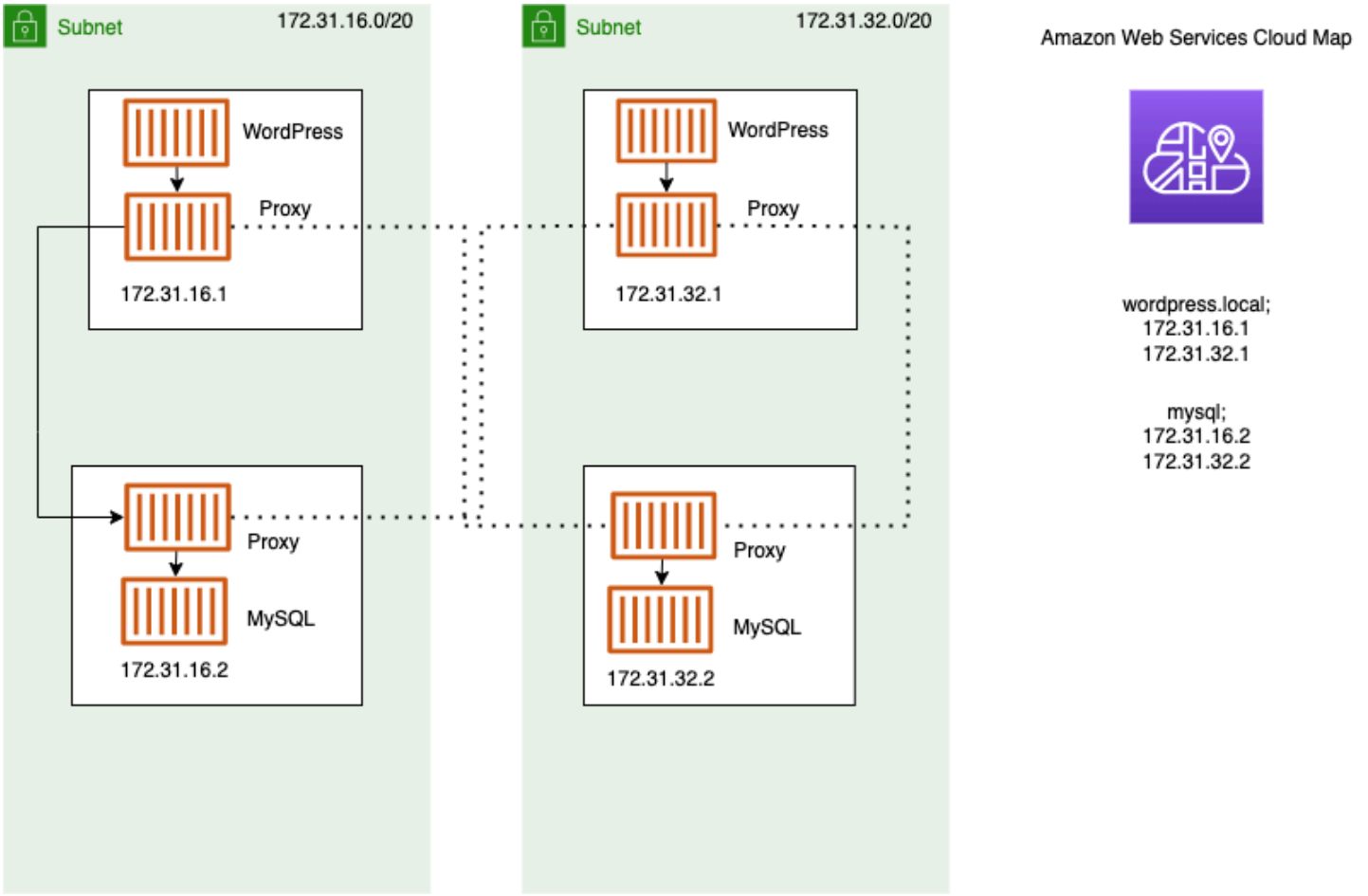
下表介绍了这些选项与任务网络模式之间的兼容性。在表中，“客户端”是指从 Amazon ECS 任务内部建立连接的应用程序。

互连选项	已进行桥接	**awsvpc**	Host
服务发现	是的，但要求客户端知道 DNS 中没有 hostPort 的 SRV 记录。	是	是的，但要求客户端知道 DNS 中没有 hostPort 的 SRV 记录。
Service Connect	是	是	否

使用 Service Connect 连接具有短名称的 Amazon ECS 服务

Amazon ECS Service Connect 以 Amazon ECS 配置的形式提供服务间通信的管理。它在 Amazon ECS 中同时构建服务发现和服务网格。这让您可以通过服务部署控制您管理的每个服务内部的全面配置，使用不依赖 VPC DNS 配置的统一方式在命名空间中引用您的服务，以及使用标准化的指标和日志来监控您的所有应用程序。Service Connect 仅会互连服务。

下图显示了一个示例 Service Connect 网络，其具有 VPC 中的 2 个子网和有 2 个服务。一种运行 WordPress 的客户端服务，其在每个子网中有 1 个任务。一种运行 MySQL 的服务器服务，其在每个子网中有 1 个任务。这两项服务都具有很高的可用性，可以弹性应对任务和可用区问题，因为每项服务都运行分布在 2 个子网中的多个任务。实心箭头会显示从 WordPress 到 MySQL 的连接。例如，使用 IP 地址 `172.31.16.1` 在任务中从 WordPress 容器内部运行的 `mysql --host=mysql` CLI 命令。该命令在 MySQL 的默认端口上使用简称 `mysql`。此名称和端口会在同一任务中连接到 Service Connect 代理。WordPress 任务中的代理使用轮询负载均衡和异常值检测中的任何先前失败信息来选择要连接到的 MySQL 任务。如图中的实心箭头所示，代理使用 IP 地址 `172.31.16.2` 连接到 MySQL 任务中的第二个代理。第二个代理会在同一任务中连接到本地 MySQL 服务器。这两个代理均会报告连接性能，这些性能在 Amazon ECS 和 Amazon CloudWatch 控制台的图表中可见，因此您可以用相同的方式从各种应用程序中获取性能指标。



以下服务可与 Service Connect 一起使用：

端口名称

为特定端口映射分配名称的 Amazon ECS 任务定义配置。此配置仅供 Amazon ECS Service Connect 使用。

客户端别名

分配端点中使用的端口号的 Amazon ECS 服务配置。此外，客户端别名可以分配端点的 DNS 名称，覆盖发现名称。如果 Amazon ECS 服务中未提供发现名称，则客户端别名将覆盖端口名称作为端点名称。有关端点示例，请参阅端点的定义。可以为一项 Amazon ECS 服务分配多个客户端别名。此配置仅供 Amazon ECS Service Connect 使用。

发现名称

可以为任务定义中的指定端口创建的可选中间名称。此名称用于创建 AWS Cloud Map 服务。如果此名称未提供，则使用任务定义中的端口名称。可以为 Amazon ECS 服务的特定端口分配多个发现名称。此配置仅供 Amazon ECS Service Connect 使用。

AWS Cloud Map 服务名称在命名空间中必须是唯一的。由于此限制，对于每个命名空间中的特定任务定义，您只能有一个 Service Connect 配置，而没有发现名称。

端点

连接到 API 或网站的 URL。URL 包含协议、DNS 名称和端口。有关一般端点的更多信息，请参阅《Amazon Web Services 一般参考》中的 AWS 词汇表中的端点。

Service Connect 创建连接到 Amazon ECS 服务的端点，并在 Amazon ECS 服务中配置任务以连接到端点。URL 包含协议、DNS 名称和端口。您可以在任务定义中选择协议和端口名称，因为该端口必须与容器映像内的应用程序匹配。在服务中，您可以按名称选择每个端口，可以分配 DNS 名称。如果您未在 Amazon ECS 服务配置中指定 DNS 名称，则默认使用任务定义中的端口名称。例如，Service Connect 端点可以是 `http://blog:80`、`grpc://checkout:8080` 或 `http://_db.production.internal:99`。

Service Connect 服务

Amazon ECS 服务中单一端点的配置。这是 Service Connect 配置的一部分，由控制台中的 Service Connect 和发现名称配置中的单行组成，或由 Amazon ECS 服务的 JSON 配置中的 `services` 列表中的一个对象组成。此配置仅供 Amazon ECS Service Connect 使用。

有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 ServiceConnectService。

命名空间

与 Service Connect 结合使用的 AWS Cloud Map 命名空间的 Amazon 资源名称（ARN）短名称或全称。命名空间必须与 Amazon ECS 服务和集群在同一个 AWS 区域 内。AWS Cloud Map 中的命名空间类型不影响 Service Connect。

Service Connect 将 AWS Cloud Map 命名空间用作相互对话的 Amazon ECS 任务的逻辑分组。每个 Amazon ECS 服务所属的命名空间只能是唯一的。命名空间中的服务可以分布在同一个 AWS 账户 中的同一个 AWS 区域 内的不同 Amazon ECS 集群中。您可以根据任何标准自由地组织服务。

客户端服务

一种运行网络客户端应用程序的服务。此服务必须配置命名空间。服务中的每项任务都可以通过 Service Connect 代理容器发现并连接到命名空间中的所有端点。

如果任务中的任何容器需要从命名空间中的服务连接到端点，请选择客户端服务。如果前端、反向代理或负载均衡器应用程序通过其他方法（例如从 Elastic Load Balancing 中）接收外部流量，则它可以使用这种类型的 Service Connect 配置。

客户端-服务器服务

一种运行网络或 Web 服务应用程序的 Amazon ECS 服务。此服务必须配置命名空间和至少一个端点。使用端点即可访问服务中的每项任务。Service Connect 代理容器侦听端点名称和端口，将流量引导到任务中的应用程序容器。

如果有任何容器公开并侦听端口上的网络流量，请选择客户端-服务器服务。这些应用程序无需连接到同一命名空间中的其他客户端-服务器服务，但需要客户端配置。后端、中间件、业务层或大多数微服务都可以使用这种类型的 Service Connect 配置。如果您希望前端、反向代理或负载均衡器应用程序接收来自在相同命名空间中的使用 Service Connect 进行配置的其他服务的流量，则这些服务应使用此类 Service Connect 配置。

Service Connect 功能可创建相关服务的虚拟网络。可以在多个不同的命名空间中使用相同的服务配置，以运行独立但相同的应用程序集。Service Connect 定义了 Amazon ECS 服务中的代理容器。通过这种方式，就可以使用相同的任务定义在具有不同 Service Connect 配置的不同命名空间中运行相同的应用程序。服务制造的每项任务都会在任务中运行一个代理容器。

Service Connect 适用于同一命名空间内的 Amazon ECS 服务之间的连接。对于以下应用程序，您需要使用其他互连方法连接到配置了 Service Connect 的 Amazon ECS 服务：

- 在其他命名空间中配置的任务
- 未针对 Service Connect 配置的任务
- Amazon ECS 之外的其他应用程序

这些应用程序可以通过 Service Connect 代理进行连接，但无法解析 Service Connect 端点名称。

若要让这些应用程序解析 Amazon ECS 任务的 IP 地址，则需使用另一种互连方法。

定价

Amazon ECS Service Connect 的定价取决于您是使用 AWS Fargate 还是 Amazon EC2 基础设施来托管您的容器化工作负载。在 AWS Outposts 上使用 Amazon ECS 时，则定价采用与您直接使用 Amazon EC2 时相同的模式。有关更多信息，请参阅 Amazon ECS 定价。

当 Service Connect 使用它时，AWS Cloud Map 使用完全免费。

Amazon ECS Service Connect 组件

使用 Amazon ECS Service Connect 时，可以将每项 Amazon ECS 服务配置为运行接收网络请求的服务器应用程序（客户端-服务器服务），或者配置为运行发出请求的客户端应用程序（客户端服务）。

准备开始使用 Service Connect 时，从客户端-服务器服务开始。您可以向新服务或现有服务添加 Service Connect 配置。Amazon ECS 会在命名空间中创建一个 Service Connect 端点。此外，Amazon ECS 在服务中创建新部署以替换当前正在运行的任务。

现有任务和其他应用程序可以继续连接到现有端点和外部应用程序。如果客户端-服务器服务通过横向扩展来添加任务，则来自客户端的新连接将在所有任务之间保持平衡。如果客户端-服务器服务已更新，则来自客户端的新连接将在新版本的任务之间保持平衡。

现有任务无法解析并连接到新端点。只有在同一命名空间中具有 Service Connect 配置且在此部署之后开始运行的新任务才能解析并连接到此端点。

这意味着客户端应用程序的操作员可以决定其应用程序的配置何时更改，即使服务器应用程序的操作员可以随时更改其配置。每次部署命名空间中的任何服务时，命名空间中的端点列表都可能发生变化。现有任务和替换任务的行为将继续与最近部署后的行为相同。

考虑以下示例。

首先，假设您正在创建可在单个 AWS CloudFormation 模板和单个 AWS CloudFormation 堆栈中供公有互联网使用的应用程序。公共发现和可访问性应最后由 AWS CloudFormation 创建，包括前端客户端服务。服务需要按此顺序创建，以防止前端客户端服务运行并向公众开放，但后端不会。这样可以避免在此期间向公众发送错误消息。在 AWS CloudFormation 中，您必须使用 `dependsOn`，用于向 AWS CloudFormation 指示多个 Amazon ECS 服务不能并行或同时进行。对于客户端任务连接到的每个后端客户端-服务器服务，您都应将 `dependsOn` 添加到前端客户端服务中。

其次，假设存在没有 Service Connect 配置的前端服务。任务正在连接到现有的后端服务。首先使用 DNS 中的相同名称或前端所用的 `clientAlias`，将客户端-服务器服务连接配置添加到后端服务。这会创建一个新部署，因此所有部署回滚检测或 AWS Management Console、AWS CLI、AWS 开发工具包和其他方法将后端服务回滚并恢复到先前的部署和配置。如果您对后端服务的性能和行为感到满意，请向前端服务添加客户端或客户端-服务器 Service Connect 配置。只有新部署中的任务使用添加到这些新任务中的 Service Connect 代理。如果此配置有问题，通过使用部署回滚检测或 AWS Management Console、AWS CLI、AWS 开发工具包和其他方法将后端服务回滚并恢复到先前的部署和配置，您可以回滚和恢复到以前的配置。如果您使用其他基于 DNS（而非 Service Connect）的服务发现系统，则在本地 DNS 缓存到期后，任何前端或客户端应用程序都会开始使用新端点并更改端点配置，这通常需要数小时。

联网

默认情况下，Service Connect 代理从任务定义端口映射中侦听 `containerPort`。您的安全组规则必须允许从运行客户端的子网传入（进入）流量到此端口。

即使您在 Service Connect 服务配置中设置了端口号，这也不会更改 Service Connect 代理侦听的客户端-服务器服务的端口。当您设置此端口号时，Amazon ECS 会在这些任务内的 Service Connect 代理上更改客户端服务所连接的端点的端口。客户端服务中的代理使用 `containerPort` 连接到客户端-服务器服务中的代理。

如果要更改 Service Connect 代理侦听的端口，请更改客户端服务器服务的 Service Connect 配置中的 `ingressPort0verride`。如果您更改此端口号，则必须允许此端口上的入站流量，该端口由通往此服务的流量使用。

您的应用程序发送到为 Service Connect 配置的 Amazon ECS 服务的流量要求 Amazon VPC 和子网具有允许您使用的 `containerPort` 和 `ingressPort0verride` 端口号的路由表规则和网络 ACL 规则。

您可以使用 Service Connect 在 VPC 之间发送流量。对路由表规则、网络 ACL 和安全组的相同要求同时适用于两种 VPC。

例如，两个集群在不同的 VPC 中创建任务。每个集群中的服务都配置为使用相同的命名空间。这两个服务中的应用程序无需任何 VPC DNS 配置即可解析命名空间中的每个端点。但是，除非 VPC 对等、VPC 或子网路由表以及 VPC 网络 ACL 允许 `containerPort` 和 `ingressPort0verride` 端口号上的流量，否则代理无法进行连接。

对于使用 `bridge` 联网模式的任务，您必须创建一个带有入站规则的安全组，该规则允许上动态端口范围内的流量。然后，将安全组分配给 Service Connect 集群中的所有 EC2 实例。

Service Connect 代理

如果使用 Service Connect 配置创建或更新服务，Amazon ECS 则会在每个新任务启动时为其添加一个新容器。这种使用单独容器的模式称为 `sidecar`。此容器不存在于任务定义中，您无法对其进行配置。Amazon ECS 在服务中管理容器配置。这允许您在多个服务、命名空间和任务之间重复使用相同的任务定义，而无需 Service Connect。

代理资源

- 对于任务定义，您必须设置 CPU 和内存参数。

  我们建议为 Service Connect 代理容器的任务 CPU 和内存添加 256 个 CPU 单元和至少 64MiB 的内存。在 AWS Fargate 上，可以设置的最低内存量为 512 MiB 内存。在 Amazon EC2 上，任务定义内存是必需的。
- 对于该服务，您可以在 Service Connect 配置中设置日志配置。

- 如果您希望此服务中的任务在峰值负载时每秒收到超过 500 个请求，我们建议在此 Service Connect 代理容器的任务定义中向您的任务 CPU 添加 512 个 CPU 单元。
- 如果您希望在命名空间中创建 100 个以上的 Service Connect 服务，或者在命名空间内的所有 Amazon ECS 服务中总共创建 2000 个任务，我们建议在 Service Connect 代理容器的任务内存中添加 128 MiB 的内存。在命名空间中的所有 Amazon ECS 服务使用的每个任务定义中，您都应该执行此操作。

代理配置

您的应用程序通过与应用程序相同的任务连接到 sidecar 容器中的代理。Amazon ECS 会配置任务和容器，以便只有当应用程序连接到同一命名空间中的端点名称时，应用程序才会连接到代理。所有其他流量都不使用此代理。其他流量包括同一 VPC 中的 IP 地址、AWS 服务端点和外部流量。

负载均衡

Service Connect 将代理配置为使用轮询策略在 Service Connect 端点中的任务之间进行负载均衡。连接所在任务中的本地代理在提供端点的客户端-服务器服务中挑选其中一个任务。

例如，考虑一个在名为 local 的命名空间中配置为客户端服务的服务中运行 WordPress 的任务。还有另一项包含运行 MySQL 数据库的 2 个任务的服务。此服务配置为在同一命名空间中通过 Service Connect 提供一个称为 `mysql` 的端点。在 WordPress 任务中，WordPress 应用程序会使用端点名称连接到数据库。与该名称的连接会转到在同一个任务中在 sidecar 容器中运行的代理。然后，代理可以使用轮询策略连接到任一 MySQL 任务。

负载均衡策略：轮询

异常检测

此功能使用代理拥有的有关先前失败连接的数据，以避免向已连接失败的主机发送新连接。Service Connect 配置代理的异常检测功能以提供被动运行状况检查。

使用上一个示例，代理可以连接到任一 MySQL 任务。如果代理与特定 MySQL 任务建立了多个连接，并且在过去 30 秒内有 5 个或以上的连接失败，则代理会在 30 到 300 秒内避开该 MySQL 任务。

重试

Service Connect 将代理配置为重试通过代理但失败的连接，第二次尝试将避免使用先前连接中的主机。这样可以确保通过 Service Connect 所进行的每次连接都不会因为一次性原因而失败。

重试次数：2

超时

Service Connect 将代理配置为客户端-服务器应用程序响应的最长等待时间。默认超时值为 15 秒，但该值可以更新。

可选参数：

idleTimeout - 连接在空闲时保持活动状态的时间（以秒为单位）。值为 0 将禁用 idleTimeout。

HTTP/HTTP2/GRPC 的 idleTimeout 默认值为 5 分钟。

TCP 的 idleTimeout 默认值为 1 小时。

perRequestTimeout – 等待上游对每个请求做出完整响应的时长。值为 0 将关闭 perRequestTimeout。仅当应用程序容器的 appProtocol 为 HTTP/HTTP2/GRPC 时才能设置此选项。默认值为 15 秒。

> **Note**
>
> 如果将 idleTimeout 设置为小于 perRequestTimeout 的时间，则连接将在达到 idleTimeout 而不是 perRequestTimeout 时关闭。

注意事项

在使用 Service Connect 时请注意以下事项：

- 在 Fargate 中运行的任务必须使用 Fargate Linux 平台版本 1.4.0 或更高版本才能使用 Service Connect。
- 容器实例上的 Amazon ECS 代理版本必须为 1.67.2 或更高版本。
- 容器实例必须运行经 Amazon ECS 优化的 Amazon Linux 2023 AMI 版本 20230428 或更高版本或者经 Amazon ECS 优化的 Amazon Linux 2 AMI 版本 2.0.20221115 才能使用 Service Connect。除了 Amazon ECS 容器代理之外，这些版本还具有 Service Connect 代理。有关 Service Connect 代理的更多信息，请参阅 GitHub 上的 Amazon ECS Service Connect 代理。
- 容器实例必须具有对资源 arn:aws:ecs:*region*:*0123456789012*:task-set/*cluster*/* 的 ecs:Poll 权限。如果您使用的是 ecsInstanceRole，则无需添加其他权限。AmazonEC2ContainerServiceforEC2Role 托管策略具有必要权限。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。
- Service Connect 只支持使用滚动部署的服务。

- 使用 `bridge` 网络模式并使用 Service Connect 的任务不支持 `hostname` 容器定义参数。
- 任务定义必须设置任务内存限制才能使用 Service Connect。有关更多信息，请参阅 Service Connect 代理。
- 不支持设置容器内存限制的任务定义。

  您可以在容器上设置容器内存限制，但必须将任务内存限制设置为大于容器内存限制总和的数字。未在容器限制中分配的任务限制中的额外 CPU 和内存将由 Service Connect 代理容器和其他未设置容器限制的容器使用。有关更多信息，请参阅 Service Connect 代理。
- 您可以将 Service Connect 配置为使用同一个 AWS 账户 中的相同区域中的任何 AWS Cloud Map 命名空间。
- 每个服务所属的命名空间只能是唯一的。
- 仅支持服务创建的任务。
- 所有端点在命名空间内必须是唯一的。
- 所有发现名称在命名空间内必须是唯一的。
- 您必须重新部署现有服务，然后应用程序才能解析新的端点。在最新部署后添加到命名空间的新端点不会添加到任务配置中。有关更多信息，请参阅 the section called “Service Connect 组件”。
- 删除集群时，Service Connect 不会删除命名空间。您必须删除 AWS Cloud Map 中的命名空间。
- 应用程序负载均衡器流量默认为在 `awsvpc` 网络模式下通过 Service Connect 代理进行路由。如果您希望非服务流量绕过 Service Connect 代理，则请在 Service Connect 服务配置中使用 `ingressPortOverride` 参数。

Service Connect 不支持以下各项：

- Windows 容器
- HTTP 1.0
- 独立任务
- 使用蓝绿和外部部署类型的服务
- Service Connect 不支持 Amazon ECS Anywhere 的 `External` 容器实例。
- PPv2

支持 Service Connect 的区域

Amazon ECS Service Connect 在以下 AWS 区域可用：

区域名称	区域
美国东部（俄亥俄州）	us-east-2
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（北加利福尼亚）	us-west-1
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（雅加达）	ap-southeast-3
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（墨尔本）	ap-southeast-4
亚太地区（东京)	ap-northeast-1
加拿大（中部）	ca-central-1
加拿大西部（卡尔加里）	ca-west-1
中国（北京）	cn-north-1（注意：Service Connect 的 TLS 不适用于该区域。）
中国（宁夏）	cn-northwest-1（注意：Service Connect 的 TLS 不适用于该区域。）

区域名称	区域
欧洲（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3
欧洲（米兰）	eu-south-1
欧洲（西班牙）	eu-south-2
欧洲地区（斯德哥尔摩）	eu-north-1
欧洲（苏黎世）	eu-central-2
以色列（特拉维夫）	il-central-1
中东（巴林）	me-south-1
中东（阿联酋）	me-central-1
南美洲（圣保罗）	sa-east-1

Amazon ECS Service Connect 配置概述

使用 Service Connect 时，需要在资源中配置一些参数。

需要为 Service Connect 配置的 Amazon ECS 资源

参数位置	应用程序类型	描述	必填
任务定义	客户端	在客户端任务定义中，Service Connect 没有可用的更改。	不适用
任务定义	客户端-服务器	服务器必须向容器的 `portMappings` 中的端口添加 `name` 字段。有关更多信息，请参阅 `portMappings`	是

参数位置	应用程序类型	描述	必填
任务定义	客户端-服务器	服务器可以选择提供应用程序协议（例如，HTTP）来接收其服务器应用程序的协议特定的指标（例如，HTTP 5xx）。	否
服务定义	客户端	客户端服务必须添加 serviceConnectConfiguration 才能配置要加入的命名空间。此命名空间必须包含该服务需要发现的所有服务器服务。有关更多信息，请参阅 serviceConnectConfiguration。	是
服务定义	客户端-服务器	服务器服务必须添加 serviceConnectConfiguration 才能配置提供该服务的 DNS 名称、端口号和命名空间。有关更多信息，请参阅 serviceConnectConfiguration。	是
集群	客户端	集群可以添加默认的 Service Connect 命名空间。在服务中配置 Service Connect 后，集群中的新服务会继承命名空间。	否
集群	客户端-服务器	在适用于服务器服务的集群中，Service Connect 没有可用的更改。服务器任务定义和服务必须设置相应的配置。	不适用

配置 Service Connect 的步骤概述

以下步骤概述了如何配置 Service Connect。

Important

- Service Connect 在您的账户中创建 AWS Cloud Map 服务。通过手动注册/取消注册实例、更改实例属性或删除服务来修改这些 AWS Cloud Map 资源可能会导致应用程序流量或后续部署出现意外行为。

- Service Connect 不支持任务定义中的链接。

1. 将端口名称添加到您的任务定义的端口映射中。此外，您可以识别应用程序的第 7 层协议，以获得更多指标。
2. 创建带有 AWS Cloud Map 命名空间的集群或单独创建命名空间。对于简单的组织，使用您想要用于命名空间的名称创建一个集群并为命名空间指定相同的名称。在这种情况下，Amazon ECS 使用必要的配置创建了一个新的 HTTP 命名空间。Service Connect 不在 Amazon Route 53 中使用或创建 DNS 托管区。
3. 配置服务以在命名空间内创建 Service Connect 端点。
4. 部署服务以创建端点。Amazon ECS 向每个任务添加一个 Service Connect 代理容器，并在 AWS Cloud Map 中创建 Service Connect 端点。此容器未在任务定义中配置，任务定义无需修改即可重复使用，以便在同一个命名空间或多个命名空间中创建多个服务。
5. 将客户端应用程序部署为服务以连接到端点。Amazon ECS 通过每项任务中的 Service Connect 代理将它们连接到 Service Connect 端点。

   应用程序仅会使用代理连接到 Service Connect 端点。无需其他配置，即可使用代理。代理执行轮询负载均衡、异常值检测和重试。有关代理的更多信息，请参阅 Service Connect 代理。
6. 通过 Amazon CloudWatch 中的 Service Connect 代理监控流量。

集群配置

您可以在创建或更新集群时设置 Service Connect 的默认命名空间。如果您指定的命名空间名称不存在于同一个 AWS 区域 和账户中，则会创建一个新的 HTTP 命名空间。

如果您创建集群并指定默认 Service Connect 命名空间，则在 Amazon ECS 创建命名空间期间，该集群将处于 `PROVISIONING` 状态等待。您可以在集群的状态中看到一个 `attachment`，它显示了命名空间的状态。默认情况下，附件不显示在 AWS CLI 中，您必须添加 `--include ATTACHMENTS` 才能查看。

服务配置

Service Connect 旨在要求最低配置。您需要为要在任务定义中与 Service Connect 结合使用的每个端口映射设置一个名称。在服务中，您需要开启 Service Connect 并选择一个命名空间来创建客户端服务。要创建客户端-服务器服务，您需要添加与其中一个端口映射的名称相匹配的单个 Service Connect 服务配置。Amazon ECS 重复使用任务定义中的端口号和端口名来定义 Service Connect 服务和端

点。要覆盖这些值，可以在控制台中使用其他参数 Discovery、DNS 和 Port，或者在 Amazon ECS API 中分别使用 `discoveryName` 和 `clientAliases`。

以下示例显示了在同一 Amazon ECS 服务中一起使用的每种 Service Connect 配置。提供了 Shell 注释，但请注意，用于 Amazon ECS 服务的 JSON 配置不支持注释。

```
{
    ...
    serviceConnectConfiguration: {
        enabled: true,
        namespace: "internal",
        #config for client services can end here, only these two parameters are
 required.
        services: [{
                portName: "http"
            }, #minimal client - server service config can end here.portName must match
 the "name"
            parameter of a port mapping in the task definition. {
                discoveryName: "http-second"
                #name the discoveryName to avoid a Task def port name collision with
 the minimal config in the same Cloud Map namespace
                portName: "http"
            },
            {
                clientAliases: [{
                        dnsName: "db",
                        port: 81
                    }] #use when the port in Task def is not the port that client apps
 use.Client apps can use http: //db:81 to connect
                    discoveryName: "http-three"
                portName: "http"
            },
            {
                clientAliases: [{
                    dnsName: "db.app",
                    port: 81
                }] #use when the port in Task def is not the port that client apps
 use.duplicates are fine as long as the discoveryName is different.
                discoveryName: "http-four"
                portName: "http",
                ingressPortOverride: 99 #If App should also accept traffic directly on
 Task def port.
```

```
            }
        ]
    }
}
```

加密 Amazon ECS Service Connect 流量

Amazon ECS Service Connect 支持使用传输层安全性协议（TLS）证书对 Amazon ECS 服务进行自动流量加密。当您将 Amazon ECS 服务指向 AWS Private Certificate Authority（AWS Private CA）时，Amazon ECS 会自动预调配 TLS 证书，以加密您的 Amazon ECS Service Connect 服务之间的流量。Amazon ECS 生成、轮换和分发用于流量加密的 TLS 证书。

通过 Service Connect 进行自动流量加密将使用业界领先的加密功能来保护您的服务间通信，从而帮助您满足安全要求。它支持使用 256-bit ECDSA 和 2048-bit RSA 加密的 AWS Private Certificate Authority TLS 证书。默认情况下，支持 TLS 1.3，但不支持 TLS 1.0 - 1.2。您还可以完全控制私有证书和签名密钥，从而帮助您满足合规性要求。

> **Note**
>
> 要使用 TLS 1.3，您必须在目标上的侦听器上启用它。
> 只有通过 Amazon ECS 代理的入站和出站流量才会被加密。

AWS Private Certificate Authority 证书和 Service Connect

颁发证书还需要其他的 IAM 权限。Amazon ECS 提供了托管式资源信任策略，其中概述了权限集。有关这项新策略的更多信息，请参阅 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity。

Service Connect 的 AWS Private Certificate Authority 模式

AWS Private Certificate Authority 可以在两种模式下运行：通用模式和短暂模式。

- 通用 – 可配置为任何到期日期的证书。
- 短期 – 最长有效期为七天的证书。

虽然 Amazon ECS 对这两种模式都支持，但建议使用短期证书。默认情况下，证书每五天轮换一次，与通用型相比，在短期模式下运行可以节省大量成本。

Service Connect 不支持证书吊销，而是利用进行频繁的证书轮换的短期证书。您有权在 Secrets Manager 中使用托管轮换修改轮换频率、禁用或删除密钥，但这样做可能会带来以下可能的后果。

- 较短的轮换频率 – 较短的轮换频率会因 AWS Private CA、AWS KMS 和 Secrets Manager 以及轮换工作负载增加的自动扩缩而导致更高的成本。
- 更长的轮换频率 – 如果轮换频率超过七天，应用程序的通信就会失败。
- 删除密钥 – 删除密钥会导致轮换失败并影响客户应用程序通信。

如果您的密钥轮换失败，则会在 AWS CloudTrail 中发布 `RotationFailed` 事件。您也可以为 `RotationFailed` 设置 CloudWatch 警报。

> **Important**
>
> 不要将副本区域添加到密钥中。这样做可以防止 Amazon ECS 删除密钥，因为 Amazon ECS 无权从复制中移除区域。如果您已经添加了复制，请运行以下命令。
>
> ```
> aws secretsmanager remove-regions-from-replication \
>  --secret-id SecretId \
>  --remove-replica-regions region-name
> ```

从属证书颁发机构

您可以将任何 AWS Private CA（根或从属）带到 Service Connect TLS，从而为服务颁发终端实体证书。所提供的颁发者在任何地方都被视为签名者和可信根。您可以从不同的从属 CA 为应用程序的不同部分颁发终端实体证书。使用 AWS CLI 时，请提供用于建立信任链的 CA 的 Amazon 资源名称（ARN）。

本地证书颁发机构

要使用您的本地 CA，请在 AWS Private Certificate Authority 中创建和配置从属 CA。这样可以确保为您的 Amazon ECS 工作负载颁发的所有 TLS 证书都与您在本地运行的工作负载共享信任链，并且能够安全连接。

> **Important**
>
> 请在您的 AWS Private CA 中添加所需的标签 `AmazonECSManaged : true`。

基础设施即代码

将 Service Connect TLS 与基础设施即代码（IaC）工具配合使用时，请务必正确配置依赖项，以避免出现服务停滞在耗尽状态等问题。使用完您的 Amazon ECS 服务后应将您的 AWS KMS 密钥（如果提供）、IAM 角色和 AWS Private CA 依赖项删除。

Service Connect 和 AWS Key Management Service

您可以使用 AWS Key Management Service 来加密和解密您的 Service Connect 资源。AWS KMS 是 AWS 管理的一项服务，可以通过该服务创建和管理用于保护数据的加密密钥。

将 AWS KMS 与 Service Connect 结合使用时，您可以选择使用 AWS 为您管理的 AWS 拥有的密钥，也可以选择现有的 AWS KMS 密钥。您也可以创建新的 AWS KMS 密钥以供使用。

提供您自己的加密密钥

您可以提供自己的密钥材料，也可以通过 AWS Key Management Service 使用外部密钥存储，将自己的密钥导入 AWS KMS，然后在 Amazon ECS Service Connect 中指定该密钥的 Amazon 资源名称（ARN）。

以下是示例 AWS KMS 策略。将所有 *user input* 值替换为您自己的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "id",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role-name"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyPair"
      ],
      "Resource": "*"
    }
  ]
}
```

有关密钥政策的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥政策。

> **Note**
>
> Service Connect 仅支持对称加密 AWS KMS 密钥。您不能使用任何其他类型的 AWS KMS 密钥来加密您的 Service Connect 资源。有关确定 AWS KMS 密钥是否为对称加密密钥的帮助，请参阅识别对称和非对称 AWS KMS 密钥。

有关 AWS Key Management Service 对称加密密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 AWS KMS 密钥。

为 Amazon ECS Service Connect 启用 TLS

您可以在创建或更新 Service Connect 服务时启用流量加密。

要使用 AWS Management Console 为现有命名空间中的服务启用流量加密

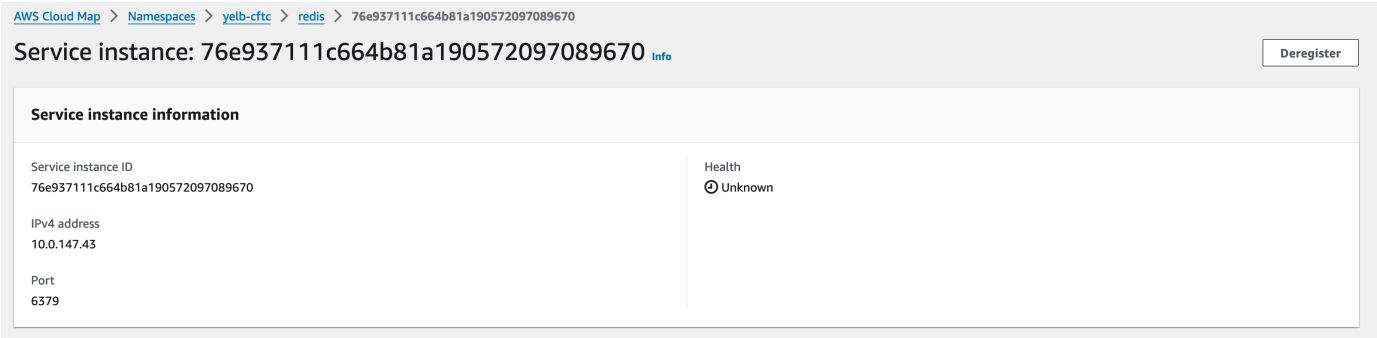
1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Namespaces (命名空间)。
3. 选择包含您要为其启用流量加密的服务的命名空间。
4. 选择您要为其启用流量加密的服务。
5. 选择右上角的更新服务，然后向下滚动到 Service Connect 部分。
6. 在服务信息下选择开启流量加密以启用 TLS。
7. 对于 Service Connect TLS 角色，选择现有角色或创建新角色。
8. 对于签名者证书颁发机构，请选择现有的证书颁发机构或创建一个新的证书颁发机构。
9. 对于选择一个 AWS KMS key，选择 AWS 拥有的密钥和托管密钥，或者可以选择其他密钥。您也可以选择创建一个新密钥。

有关使用 AWS CLI 为您的服务 使用 AWS CLI 配置 Amazon ECS Service Connect 配置 TLS 的示例。

验证 Amazon ECS Service Connect 是否已启用 TLS

Service Connect 在 Service Connect 代理处启动 TLS，并在目标代理处将其终止。因此，应用程序代码永远看不到 TLS 交互。使用以下步骤验证 TLS 是否已启用。

1. 确保您的应用程序映像具有 openssl CLI。
2. 在您的服务上启用 ECS Exec 以通过 SSM 连接到您的任务。或者，您可以在与服务相同的 Amazon VPC 中启动 Amazon EC2 实例。
3. 从要验证的服务中检索任务的 IP 和端口。例如，如果您的 redis 服务已开启 TLS，则可以通过导航到 AWS Cloud Map、查找服务并查看一个实例的 IP 和端口来检索其任务 IP。



4. 使用如下例所示的 execute-command 登录您的任何任务。或者，登录到步骤 2 中创建的 Amazon EC2 实例。

```
$ aws ecs execute-command --cluster cluster-name \
    --task < TASK_ID>  \
    --container app \
    --interactive \
    --command "/bin/sh"
```

> **Note**
>
> 直接调用 DNS 名称不会显示证书。

5. 在连接的 shell 中，使用 openssl CLI 验证和查看附加到任务的证书。

   例如：

```
openssl s_client -connect 10.0.147.43:6379 < /dev/null 2> /dev/null \
| openssl x509 -noout -text
```

   响应示例：

```
Certificate:
    Data:
        Version: 3 (0x2)
```

```
        Serial Number:
            <serial-number>
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: <issuer>
        Validity
            Not Before: Jan 23 21:38:12 2024 GMT
            Not After : Jan 30 22:38:12 2024 GMT
        Subject: <subject>
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    <pub>
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:redis.yelb-cftc
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:<key-id>

            X509v3 Subject Key Identifier:
                1D:<id>
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
    Signature Algorithm: ecdsa-with-SHA256
        <hash>
```

使用 AWS CLI 配置 Amazon ECS Service Connect

您可以创建 Fargate 任务的 Amazon ECS 服务，该任务将 Service Connect 和 AWS CLI 结合使用。

先决条件

以下是 Service Connect 的先决条件：

- 验证该区域是否支持 Service Connect。有关更多信息，请参阅 Regions with Service Connect。

- 验证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 Command Line Interface。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 您已创建要使用的 VPC、子网、路由表和安全组。有关更多信息，请参阅 the section called “创建 Virtual Private Cloud”。
- 您有一个名为 `ecsTaskExecutionRole` 的任务执行角色，并且 `AmazonECSTaskExecutionRolePolicy` 托管策略已附加到该角色。此角色允许 Fargate 将 NGINX 应用程序日志和 Service Connect 代理日志写入 Amazon CloudWatch Logs。有关更多信息，请参阅 创建任务执行 角色。

第 1 步：创建集群

请按照以下步骤创建 Amazon ECS 集群和命名空间。

要创建 Amazon ECS 集群和 AWS Cloud Map 命名空间。

1. 创建要使用的名为 `tutorial` 的 Amazon ECS 集群。参数 `--service-connect-defaults` 设置集群的默认命名空间。在示例输出中，此账户和 AWS 区域 中不存在名称 `service-connect` 的 AWS Cloud Map 命名空间，因此命名空间由 Amazon ECS 创建。命名空间是在账户中的 AWS Cloud Map 中创建的，所有其他命名空间都可见，因此请使用表明目的的名称。

```
aws ecs create-cluster --cluster-name tutorial --service-connect-defaults
 namespace=service-connect
```

输出：

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "clusterName": "tutorial",
        "serviceConnectDefaults": {
            "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
        },
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
```

```
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": [],
        "attachments": [
            {
                "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "type": "sc",
                "status": "ATTACHING",
                "details": []
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
}
}
```

2. 验证集群是否已创建：

```
aws ecs describe-clusters --clusters tutorial
```

输出：

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
            "clusterName": "tutorial",
            "serviceConnectDefaults": {
                "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
            },
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
```

```
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

3. （可选）验证命名空间是否是在 AWS Cloud Map 中创建的。您可以使用在 AWS Cloud Map 中创建的 AWS Management Console 或普通 AWS CLI 配置。

   例如，使用 AWS CLI：

   ```
   aws servicediscovery --region us-west-2 get-namespace --id ns-EXAMPLE
   ```

   输出：

   ```
   {
       "Namespace": {
           "Id": "ns-EXAMPLE",
           "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
   EXAMPLE",
           "Name": "service-connect",
           "Type": "HTTP",
           "Properties": {
               "DnsProperties": {
                   "SOA": {}
               },
               "HttpProperties": {
                   "HttpName": "service-connect"
               }
           },
           "CreateDate": 1661749852.422,
           "CreatorRequestId": "service-connect"
       }
   }
   ```

步骤 2：为服务器创建服务

Service Connect 功能旨在互连 Amazon ECS 上的多个应用程序。这些应用程序中至少有一个需要提供 Web 服务才能连接。在此步骤中，您将创建：

- 使用未经修改的官方 NGINX 容器映像并包括 Service Connect 配置的任务定义。
- Amazon ECS 服务定义，用于配置 Service Connect，从而为该服务的流量提供服务发现和服务网格代理。该配置重复使用集群配置中的默认命名空间，以减少您为每项服务所做的服务配置量。
- Amazon ECS 服务。它使用任务定义运行一项任务，并为 Service Connect 代理插入一个额外的容器。代理侦听任务定义的容器端口映射中的端口。在 Amazon ECS 中运行的客户端应用程序中，客户端任务中的代理侦听与任务定义端口名、服务发现名称或服务客户端别名以及来自客户端别名的端口号的出站连接。

使用 Service Connect 创建 Web 服务

1. 注册与 Fargate 兼容的任务定义并使用 `awsvpc` 网络模式。按照以下步骤进行操作：

   a. 使用以下任务定义的内容创建名为 `service-connect-nginx.json` 的文件。

      此任务定义通过向端口映射添加 `name` 和 `appProtocol` 参数来配置 Service Connect。使用多个端口时，端口名称使该端口在服务配置中更易于识别。默认情况下，端口名也用作命名空间中其他应用程序使用的可发现名称。

      任务定义包含任务 IAM 角色，因为该服务已启用 ECS Exec。

      > **Important**
      >
      > 此任务定义使用 `logConfiguration` 从 `stdout` 和 `stderr` 向 Amazon CloudWatch Logs 发送 nginx 输出。此任务执行角色不具有创建 CloudWatch Logs 日志组所需的额外权限。使用 AWS Management Console 或 AWS CLI 在 CloudWatch Logs 中创建日志组。如果您不想将 nginx 日志发送到 CloudWatch Logs，可以删除 `logConfiguration`。
      > 用您的 AWS 账户 ID 替换任务执行角色中的 AWS 账户 ID。

```
{
    "family": "service-connect-nginx",
```

```
    "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
        "name": "webserver",
        "image": "public.ecr.aws/docker/library/nginx:latest",
        "cpu": 100,
        "portMappings": [
            {
                "name": "nginx",
                "containerPort": 80,
                "protocol": "tcp",
                "appProtocol": "http"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "nginx"
            }
        }
        }
    ],
    "cpu": "256",
    "memory": "512"
}
```

b. 使用 service-connect-nginx.json 文件注册任务定义：

```
aws ecs register-task-definition --cli-input-json file://service-connect-nginx.json
```

2. 创建服务：

a. 使用您将要创建的 Amazon ECS 服务的内容，创建名为 service-connect-nginx-service.json 的文件。此示例会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。

创建 ECS 服务时，请指定 Fargate 启动类型和支持 Service Connect 的 LATEST 平台版本。securityGroups 和 subnets 必须属于符合使用 Amazon ECS 要求的 VPC。您可以从 Amazon VPC 控制台获取安全组和子网 ID。

此服务通过添加 serviceConnectConfiguration 参数来配置 Service Connect。不需要命名空间，因为集群配置了默认命名空间。在命名空间中的 ECS 中运行的客户端应用程序通过使用 portName 和 clientAliases 中的端口连接到此服务。例如，可使用 http://nginx:80/ 访问此服务，因为 nginx 在根位置 / 提供了欢迎页面。不在 Amazon ECS 中运行或不在同一命名空间中的外部应用程序可以使用任务的 IP 地址和任务定义中的端口号，通过 Service Connect 代理访问此应用程序。对于您的 tls 配置，请为您的 IAM 角色的 awsPcaAuthorityArn、kmsKey 和 roleArn 添加证书 arn。

此服务使用 logConfiguration 将 Service Connect 代理输出从 stdout 和 stderr 发送到 Amazon CloudWatch Logs。此任务执行角色不具有创建 CloudWatch Logs 日志组所需的额外权限。使用 AWS Management Console 或 AWS CLI 在 CloudWatch Logs 中创建日志组。我们建议您创建此日志组并将代理日志存储在 CloudWatch Logs 中。如果您不想将代理日志发送到 CloudWatch Logs，可以删除 logConfiguration。

```
{
    "cluster": "tutorial",
    "deploymentConfiguration": {
        "maximumPercent": 200,
        "minimumHealthyPercent": 0
    },
    "deploymentController": {
        "type": "ECS"
    },
    "desiredCount": 1,
    "enableECSManagedTags": true,
    "enableExecuteCommand": true,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "assignPublicIp": "ENABLED",
            "securityGroups": [
                "sg-EXAMPLE"
            ],
            "subnets": [
                "subnet-EXAMPLE",
```

```
                "subnet-EXAMPLE",
                "subnet-EXAMPLE"
            ]
           }
    },
    "platformVersion": "LATEST",
    "propagateTags": "SERVICE",
    "serviceName": "service-connect-nginx-service",
    "serviceConnectConfiguration": {
        "enabled": true,
        "services": [
            {
                "portName": "nginx",
                "clientAliases": [
                    {
                        "port": 80
                    }
                ],
                "tls": {
                   "issuerCertificateAuthority": {
                      "awsPcaAuthorityArn": "certificateArn"
                   },
                   "kmsKey": "kmsKey",
                   "roleArn": "iamRoleArn"
                }
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "service-connect-proxy"
            }
        }
    },
    "taskDefinition": "service-connect-nginx"
}
```

b. 使用 service-connect-nginx-service.json 文件创建服务：

```
aws ecs create-service --cluster tutorial --cli-input-json file://service-
connect-nginx-service.json
```

输出：

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/tutorial/
service-connect-nginx-service",
        "serviceName": "service-connect-nginx-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
service-connect-nginx:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 0
        },
        "deployments": [
            {
                "id": "ecs-svc/3763308422771520962",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/service-connect-nginx:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": 1661210032.602,
```

```
                "updatedAt": 1661210032.602,
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "assignPublicIp": "ENABLED",
                        "securityGroups": [
                            "sg-EXAMPLE"
                        ],
                        "subnets": [
                            "subnet-EXAMPLEf",
                            "subnet-EXAMPLE",
                            "subnet-EXAMPLE"
                        ]
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-
svc/3763308422771520962 in progress.",
                "failedLaunchTaskCount": 0,
                "replacedTaskCount": 0,
                "serviceConnectConfiguration": {
                    "enabled": true,
                    "namespace": "service-connect",
                    "services": [
                        {
                            "portName": "nginx",
                            "clientAliases": [
                                {
                                    "port": 80
                                }
                            ]
                        }
                    ],
                    "logConfiguration": {
                        "logDriver": "awslogs",
                        "options": {
                            "awslogs-group": "/ecs/service-connect-proxy",
                            "awslogs-region": "us-west-2",
                            "awslogs-stream-prefix": "service-connect-proxy"
                        },
                        "secretOptions": []
                    }
```

```
                },
                "serviceConnectResources": [
                    {
                        "discoveryName": "nginx",
                        "discoveryArn": "arn:aws:servicediscovery:us-
west-2:123456789012:service/srv-EXAMPLE"
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "version": 0,
        "events": [],
        "createdAt": 1661210032.602,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "assignPublicIp": "ENABLED",
                "securityGroups": [
                    "sg-EXAMPLE"
                ],
                "subnets": [
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE"
                ]
            }
        },
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": true,
        "propagateTags": "SERVICE",
        "enableExecuteCommand": true
    }
}
```

您提供的 serviceConnectConfiguration 出现在输出的第一个部署中。当您以需要更改任务的方式更改 ECS 服务时，Amazon ECS 会创建新的部署。

步骤 3：验证是否可以连接

要验证 Service Connect 是否已配置并正常运行，请按照以下步骤从外部应用程序连接到 Web 服务。然后，查看 CloudWatch 中 Service Connect 代理创建的其他指标。

从外部应用程序连接到 Web 服务

- 使用任务 IP 地址连接到任务 IP 地址和容器端口

  使用 AWS CLI 并利用 `aws ecs list-tasks --cluster tutorial` 获取任务 ID。

  如果您的子网和安全组允许来自任务定义的端口上的公共互联网流量，则可以从计算机连接到公有 IP。但是“describe-tasks”无法获得公有 IP，因此这些步骤包括前往 Amazon EC2 AWS Management Console 或 AWS CLI 获取弹性网络接口的详细信息。

  在此示例中，同一 VPC 中的 Amazon EC2 实例使用任务的私有 IP。应用程序是 nginx，但 `server: envoy` 标头显示使用了 Service Connect 代理。Service Connect 代理正在从任务定义侦听容器端口。

```
$ curl -v 10.0.19.50:80/
*   Trying 10.0.19.50:80...
* Connected to 10.0.19.50 (10.0.19.50) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.0.19.50
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< server: envoy
< date: Tue, 23 Aug 2022 03:53:06 GMT
< content-type: text/html
< content-length: 612
< last-modified: Tue, 16 Apr 2019 13:08:19 GMT
< etag: "5cb5d3c3-264"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 0
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

```
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

查看 Service Connect 指标

Service Connect 代理在 CloudWatch 指标中创建应用程序（HTTP、HTTP2、gRPC 或 TCP 连接）指标。使用 CloudWatch 控制台时，请查看 Amazon ECS 命名空间下的 DiscoveryName、（DiscoveryName、ServiceName、ClusterName）、TargetDiscoveryName 和（TargetDiscoveryName、ServiceName、ClusterName）等其他指标维度。有关这些指标和维度的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的查看可用指标。

使用服务发现连接具有 DNS 名称的 Amazon ECS 服务

您的 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，用于托管 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。

服务发现在以下 AWS 区域可用：

区域名称	区域
美国东部（弗吉尼亚州北部）	us-east-1
美国东部（俄亥俄州）	us-east-2
美国西部（北加利福尼亚）	us-west-1
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（孟买）	ap-south-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（东京)	ap-northeast-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Osaka)	ap-northeast-3
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（墨尔本）	ap-southeast-4
加拿大（中部）	ca-central-1
加拿大西部（卡尔加里）	ca-west-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1
欧洲（法兰克福）	eu-central-1

区域名称	区域
欧洲（苏黎世）	eu-central-2
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3
欧洲（米兰）	eu-south-1
欧洲地区（斯德哥尔摩）	eu-north-1
以色列（特拉维夫）	il-central-1
欧洲（西班牙）	eu-south-2
中东（阿联酋）	me-central-1
中东（巴林）	me-south-1
南美洲（圣保罗）	sa-east-1
AWS GovCloud（美国东部）	us-gov-east-1
AWS GovCloud（美国西部）	us-gov-west-1

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的逻辑组，例如 `example.com`。此域名是您要向其路由流量的域名。您可以通过调用 `aws servicediscovery create-private-dns-namespace` 命令或在 Amazon ECS 控制台中创建命名空间。您可以使用 `aws servicediscovery list-namespaces` 命令来查看有关当前账户创建的命名空间的摘要信息。有关服务发现命令的更多信息，请参阅 *create-private-dns-namespace*（服务发现）*list-namespaces* 参考指南中的 AWS Cloud Map 和 AWS CLI。
- 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：

- Fargate 上使用平台版本1.1.0或更高版本的任务支持服务发现。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。
- 配置为使用服务发现的服务每个服务限制为 1,000 个任务。这是由于 Route 53 服务配额造成的。
- 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。
- 必须配置 VPC DNS 属性才能成功解析 DNS。有关如何配置属性的信息，请参阅 Amazon VPC 用户指南中 VPC 中的 DNS 支持。
- 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）。
- 如果服务任务定义使用 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。
- 如果服务任务定义使用 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 discovery service name>.<service discovery namespace>。
- 在服务名称上执行 DNS 查询时，A 记录会返回一组与任务对应的 IP 地址。SRV 记录会返回每个任务的一组 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服务配置服务发现，但服务发现流量会始终路由至此任务而不会路由至负载均衡器。
- 服务发现不支持使用 Classic Load Balancers。
- 我们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您的任务状态，通过 AWS Cloud Map 更新运行状况。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。
- 使用服务发现时创建的 AWS Cloud Map 资源必须手动清理。
- 在容器运行状况检查返回值之前，任务和实例将注册为 UNHEALTHY。如果运行状况检查通过，则状态更新为 HEALTHY。如果容器运行状况检查失败，则服务发现实例将被注销。

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。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，则需要为这些运行状况检查付费。

创建使用服务发现的 Amazon ECS 服务

了解如何创建包含 Fargate 任务的务，该任务将服务发现和 AWS CLI 结合使用。

有关支持服务发现的 AWS 区域 的列表，请参阅 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。

有关支持 Fargate 的区域的信息，请参阅 the section called “AWS Fargate 区域”。

先决条件

在开始本教程之前，请确保满足以下先决条件：

- 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 Command Line Interface。
- 完成 设置以使用 Amazon ECS 中所述的步骤。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 您至少创建了一个 VPC 和一个安全组。有关更多信息，请参阅 the section called “创建 Virtual Private Cloud”。

步骤 1：在 AWS Cloud Map 中创建 Service Discovery 资源

按照以下步骤创建服务发现命名空间和服务发现服务：

1. 创建一个私有 Cloud Map 服务发现命名空间。此示例会创建一个名为 `tutorial` 的命名空间。将 *`vpc-abcd1234`* 替换为现有 VPC 之一的 ID。

```
aws servicediscovery create-private-dns-namespace \
      --name tutorial \
      --vpc vpc-abcd1234
```

   此命令的输出如下。

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. 使用上一步输出中的 `OperationId`，验证私有命名空间是否已经成功创建。记下命名空间 ID，因为您在后续命令中会使用它。

```
aws servicediscovery get-operation \
      --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e
```

   输出如下所示。

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
}
```

3. 使用上一步输出中的 NAMESPACE ID 创建服务发现服务。此示例创建一个名为 myapplication 的服务。记下服务 ID 和 ARN，因为您会在后续命令中使用它们。

```
aws servicediscovery create-service \
      --name myapplication \
      --dns-config "NamespaceId="ns-
uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]" \
      --health-check-custom-config FailureThreshold=1
```

输出如下所示。

```
{
    "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：

1. 创建 Amazon ECS 集群。此示例会创建一个名为 `tutorial` 的集群。

```
aws ecs create-cluster \
      --cluster-name tutorial
```

2. 注册与 Fargate 兼容的任务定义并使用 `awsvpc` 网络模式。按照以下步骤进行操作：

   a. 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"
}
```

b. 使用 fargate-task.json 注册任务定义。

```
aws ecs register-task-definition \
      --cli-input-json file://fargate-task.json
```

3. 按照以下步骤创建 ECS 服务：

a. 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。此示例会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。

创建 ECS 服务时，请指定 Fargate 启动类型和支持服务发现的 LATEST 平台版本。在 AWS Cloud Map 中创建服务发现服务时，registryArn 是返回的 ARN。securityGroups 和 subnets 必须属于用于创建 Cloud Map 命名空间的 VPC。您可以从 Amazon VPC 控制台获取安全组和子网 ID。

```
{
    "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
}
```

b. 使用 `ecs-service-discovery.json` 创建 ECS 服务。

```
aws ecs create-service \
      --cli-input-json file://ecs-service-discovery.json
```

步骤 3：验证 AWS Cloud Map 中的 Service Discovery

您可以通过查询您的服务发现信息来验证所有内容是否已正确创建。配置服务发现后，您可以使用 AWS Cloud Map API 操作或从 VPC 内的实例调用 `dig`。按照以下步骤进行操作：

1. 通过使用服务发现服务 ID，列出服务发现实例。记下用于资源清理的实例 ID（以粗体标记）。

```
aws servicediscovery list-instances \
      --service-id srv-utcrh6wavdkggqtk
```

输出如下所示。

```
{
    "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. 使用服务发现命名空间、服务和 ECS 集群名称等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
```

3. 在 Route 53 托管区域中为服务发现服务创建的 DNS 记录可使用以下 AWS CLI 命令进行查询：

   a. 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
```

   输出如下所示。

```
{
    "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"
    }
}
```

   b. 使用上一步中的 Route 53 托管区域 ID（请参阅粗体文本），获取托管区域的资源记录集。

```
aws route53 list-resource-record-sets \
      --hosted-zone-id Z35JQ4ZFDRYPLV
```

4. 您还可以使用 dig 从 VPC 中的实例查询 DNS。

```
dig +short myapplication.tutorial
```

步骤 4：清除

完成本教程后，清除相关资源，以避免因未使用的资源产生费用。按照以下步骤进行操作：

1. 使用您之前记下的服务 ID 和实例 ID，注销服务发现服务实例。

```
aws servicediscovery deregister-instance \
      --service-id srv-utcrh6wavdkggqtk \
      --instance-id 16becc26-8558-4af1-9fbd-f81be062a266
```

输出如下所示。

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. 使用上一步输出中的 OperationId，验证服务发现服务实例是否已成功注消。

```
aws servicediscovery get-operation \
      --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv
```

```
{
  "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. 使用服务 ID 删除服务发现服务。

```
aws servicediscovery delete-service \
      --id srv-utcrh6wavdkggqtk
```

4. 使用命名空间 ID 删除服务发现命名空间。

```
aws servicediscovery delete-namespace \
      --id ns-uejictsjen2i4eeg
```

输出如下所示。

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. 使用上一步输出中的 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
```

输出如下所示。

```
{
    "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. 将 Amazon ECS 服务的预期数量更新为 0。您必须执行此操作才能在下一步中删除该服务。

```
aws ecs update-service \
      --cluster tutorial \
      --service ecs-service-discovery \
      --desired-count 0
```

7. 删除 Amazon ECS 服务。

```
aws ecs delete-service \
```

```
      --cluster tutorial \
      --service ecs-service-discovery
```

8. 删除 Amazon ECS 集群。

```
aws ecs delete-cluster \
      --cluster tutorial
```

保护您的 Amazon ECS 任务不被横向缩减事件终止

您可以使用 Amazon ECS 任务缩容保护来保护任务，防止任务因服务自动扩缩或部署导致的横向缩减事件而终止。

某些应用程序需要一种机制来保护任务关键型任务不会在利用率低的时段或服务部署期间因横向缩减事件而终止。例如：

- 您有一个队列处理异步应用程序，例如视频转码任务，即使累积服务利用率很低，其中某些任务也需要运行数小时。
- 您的游戏应用程序将游戏服务器作为 Amazon ECS 任务运行，即使所有用户都已注销，该任务也需要继续运行，以减少服务器重启的启动延迟。
- 部署新的代码版本时，需要任务才能继续运行，因为重新处理会很昂贵。

若要防止属于您的服务的任务在横向缩减事件中终止，请将该 `protectionEnabled` 属性设置为 `true`。默认情况下，任务的保护期为 2 小时。您可以使用 `expiresInMinutes` 属性自定义保护期限。您可以保护您的任务最少 1 分钟，最多 2880 分钟（48 小时）。

任务完成其必要工作后，您可以将该 `protectionEnabled` 属性设置为 `false`，从而允许任务因随后的横向缩减事件而终止。

任务横向缩减保护机制

您可以使用 Amazon ECS 容器代理端点或 Amazon ECS API 设置和获取任务横向缩减保护。

- Amazon ECS 容器代理端点

  对于那些可以自行确定是否需要保护的任务，我们建议使用 Amazon ECS 容器代理端点。将此方法用于基于队列或任务处理的工作负载。

当容器开始处理工作时（例如，通过使用 SQS 消息），您可以通过容器内的任务横向缩减保护端点路径 `$ECS_AGENT_URI/task-protection/v1/state` 设置 `ProtectionEnabled` 属性。在横向缩减事件期间，Amazon ECS 不会终止此任务。当您的任务完成它的工作后，您可以使用相同的端点清除 `ProtectionEnabled` 属性，从而使任务有资格在随后的横向缩减事件期间终止。

有关 Amazon ECS 容器代理端点的更多信息，请参阅Amazon ECS 任务缩容保护端点。

- Amazon ECS API

  如果您的应用程序具有跟踪活动任务状态的组件，则可以使用 Amazon ECS API 来设置和检索任务缩容保护。使用 `UpdateTaskProtection` 将一个或多个任务标记为受保护。使用 `GetTaskProtection` 检索保护状态。

  这种方法的示例是，如果您的应用程序将游戏服务器会话作为 Amazon ECS 任务托管。当用户登录到服务器上的会话（任务）时，您可以将该任务标记为受保护。用户注销后，您可以清除专门针对此任务的保护，也可以定期清除对不再处于活动会话状态的类似任务的保护，具体取决于您保留空闲服务器的要求。

  有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 UpdateTaskProtection 和 GetTaskProtection。

您可以将这两种方法结合起来。例如，使用 Amazon ECS 代理端点从容器内设置任务保护，并使用 Amazon ECS API 移除对外部控制器服务的任务保护。

注意事项

在使用任务横向缩减保护之前，请注意以下几点：

- 建议使用 Amazon ECS 容器代理端点，因为 Amazon ECS 代理具有内置的重试机制和更简单的接口。
- 您可以通过在已开启保护的任务上调用 `UpdateTaskProtection` 来重置任务横向缩减保护到期时间。
- 确定任务需要多长时间才能完成其必要工作，并相应地设置 `expiresInMinutes` 属性。如果您设置的保护到期时间超过必要的时间，则将产生成本，并在部署新任务时面临延迟。
- Amazon ECS 容器代理 `1.65.0` 或更高版本支持任务横向缩减保护。

  通过将代理更新为最新版本，您可以使用较旧版本的 Amazon ECS 容器代理在 Amazon EC2 实例上添加对此功能的支持。有关更多信息，请参阅 更新 Amazon ECS 容器代理。

- 部署注意事项：
  - 如果服务使用滚动更新，则将创建新任务，但运行旧版本的任务要等到 protectionEnabled 清除或过期后才会终止。您可以将部署配置中的 maximumPercentage 参数调整为允许在保护旧任务时创建新任务的值。
  - 如果应用了蓝绿更新，则包含受保护任务的蓝色部署不会被移除（如果任务有 protectionEnabled）。流量将转移到即将出现的新任务，而较旧的任务只有在 protectionEnabled 被清除或过期时才会被删除。根据 CodeDeploy 或 CloudFormation 更新的超时时间，部署可能会超时，较旧的蓝色任务可能仍然存在。
  - 如果您使用 CloudFormation，则更新堆栈有 3 小时的超时时间。因此，如果您将任务保护设置为 3 小时以上，则您的 CloudFormation 部署可能会导致故障和回滚。

    在您的旧任务受到保护期间，CloudFormation 堆栈会显示 UPDATE_IN_PROGRESS。如果任务横向缩减保护被移除或在 3 小时内过期，则您的部署将成功并变为 UPDATE_COMPLETE 状态。如果部署在 UPDATE_IN_PROGRESS 状态停滞超过 3 个小时，将会失败并显示 UPDATE_FAILED 状态，然后将回滚到旧任务集。
  - 如果受保护的任务使部署（滚动或蓝绿）无法达到稳定状态，Amazon ECS 将发送服务事件，以便您可以采取补救措施。在尝试更新任务的保护状态时，如果您收到 DEPLOYMENT_BLOCKED 错误消息，则表示该服务的受保护任务数量超过了该服务所需的任务数。要纠正这个错误，可以执行下列操作：
    - 等待当前的任务保护过期。然后设置任务保护。
    - 确定哪些任务可以停止。然后，对这些任务使用 UpdateTaskProtection 并将 protectionEnabled 选项设置为 false。
    - 增加服务的所需任务计数，以大于受保护任务的数量。

任务横向缩减保护所需的 IAM 权限

该任务必须拥有具有以下权限的 Amazon ECS 任务角色：

- ecs:GetTaskProtection：允许 Amazon ECS 容器代理调用 GetTaskProtection。
- ecs:UpdateTaskProtection：允许 Amazon ECS 容器代理调用 UpdateTaskProtection。

Amazon ECS 任务缩容保护端点

Amazon ECS 容器代理自动将 ECS_AGENT_URI 环境变量注入 Amazon ECS 任务的容器中，以提供与容器代理 API 端点交互的方法。

对于那些可以自行确定是否需要保护的任务，我们建议使用 Amazon ECS 容器代理端点。

当容器开始处理工作时，您可以使用容器内的任务缩容保护端点路径 $ECS_AGENT_URI/task-protection/v1/state 设置 protectionEnabled 属性。

使用从容器内向此 URI 发出的 PUT 请求以设置任务缩容保护。向此 URI 发出的 GET 请求将返回任务的当前保护状态。

任务缩容保护请求参数

您可以使用具有以下请求参数的 ${ECS_AGENT_URI}/task-protection/v1/state 端点设置任务缩容保护。

ProtectionEnabled

指定 true 以标记一个保护任务。指定 false 以移除保护并使任务符合终止条件。

类型：布尔值

必需：是

ExpiresInMinutes

任务受保护的分钟数。您可以指定最少 1 分钟到最多 2,880 分钟（48 小时）。在这段时间内，您的任务不会因来自服务自动扩缩或部署的横向缩减事件而终止。在这段时间过后，protectionEnabled 参数将设置为 false。

如果您未指定时间，则任务将自动保护 120 分钟（2 小时）。

类型：整数

必需：否

以下示例显示了如何设置不同持续时间的任务保护。

如何在默认时间段内保护任务的示例

此示例显示了如何在 2 小时的默认时间段内保护任务。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-protection/v1/state --data '{"ProtectionEnabled":true}'
```

如何保护任务 60 分钟的示例

此示例显示了如何使用 expiresInMinutes 参数保护任务 60 分钟。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":60}'
```

如何保护任务 24 小时的示例

此示例显示了如何使用 expiresInMinutes 参数保护任务 24 小时。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}'
```

PUT 请求返回以下响应。

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

任务缩容保护响应参数

以下信息返回自 JSON 响应中的任务缩容保护端点 ${ECS_AGENT_URI}/task-protection/v1/state。

ExpirationDate

任务保护到期的纪元时间。如果任务未受到保护，则此值为 null。

ProtectionEnabled

任务的保护状态。如果为任务启用了横向缩减保护，则值为 true。否则为 false。

TaskArn

容器所属的任务的完整 Amazon Resource Name (ARN)。

以下示例显示了针对受保护的任务返回的详细信息。

```
curl --request GET ${ECS_AGENT_URI}/task-protection/v1/state
```

```
{
    "protection":{
        "ExpirationDate":"2023-12-20T21:57:44Z",
        "ProtectionEnabled":true,
        "TaskArn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
    }
}
```

发生故障时会返回以下信息。

`Arn`

任务的 Amazon 资源名称（ARN）。

`Detail`

与故障相关的详细信息。

`Reason`

失败的原因。

以下示例显示了针对未受保护的任务返回的详细信息。

```
{
    "failure":{
        "Arn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0",
        "Detail":null,
        "Reason":"TASK_NOT_VALID"
    }
}
```

出现异常时会返回以下信息。

`requestID`

导致异常的 Amazon ECS API 调用的 AWS 请求 ID。

`Arn`

任务或服务的 Amazon 资源名称（ARN）全名。

`Code`

错误代码。

`Message`

错误消息。

> **Note**
>
> 如果出现 `RequestError` 或 `RequestTimeout` 错误，则很可能遇到了联网问题。尝试用于 Amazon ECS 的 VPC 端点。

以下示例显示了发生错误时返回的详细信息。

```
{
    "requestID":"12345-abc-6789-0123-abc",
    "error":{
        "Arn":"arn:aws:ecs:us-west-2:555555555555:task/my-cluster-
name/1234567890abcdef0",
        "Code":"AccessDeniedException",
        "Message":"User: arn:aws:sts::444455556666:assumed-role/my-ecs-task-
role/1234567890abcdef0 is not authorized to perform: ecs:GetTaskProtection on resource:
 arn:aws:ecs:us-west-2:555555555555:task/test/1234567890abcdef0 because no identity-
based policy allows the ecs:GetTaskProtection action"
    }
}
```

如果 Amazon ECS 代理由于网络问题或 Amazon ECS 控制面板关闭等原因无法从 Amazon ECS 端点获得响应，则会出现以下错误。

```
{
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "RequestCanceled",
    "Message": "Timed out calling Amazon ECS Task Protection API"
  }
}
```

当 Amazon ECS 代理收到来自 Amazon ECS 的节流异常时，就会出现以下错误。

```
{
  "requestID": "12345-abc-6789-0123-abc",
  "error": {
```

```
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "ThrottlingException",
    "Message": "Rate exceeded"
  }
}
```

Amazon ECS 服务节流逻辑

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。

如果某个服务的任务始终无法进入 RUNNING 状态（直接从 PENDING 跳到 STOPPED 状态），则后续重启尝试间隔的时间会逐渐延长，最多达到 27 分钟。此最长周期将来可能会发生变化。此行为降低了失败任务对 Amazon ECS 集群资源或 Fargate 基础设施成本的不利影响。如果您的服务启动了限制逻辑，则您会收到以下服务事件消息：

```
(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。除了延长重启的间隔时间外，它也不会尝试以任何方式对其进行修改。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 使用控制台更新 Amazon ECS 服务。

以下是触发此逻辑的一些常见原因。建议您采取手动措施来解决此问题：

- 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。
- Amazon ECS 容器代理无法提取任务 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CannotPullContainerError停止的任务错误中看到。
- 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CannotCreateContainerError停止的任务错误中看到。有关更多信息，请参阅 对 Amazon ECS 中的 Docker API error (500): devmapper 进行故障排除。

> **Important**
>
> 进入 RUNNING 状态后停止的任务不会启动限制逻辑或相关服务事件消息。例如，假设因服务的 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，则

Amazon ECS 会将其注销并停止该任务。此时，任务没有受到限制。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导致限制或服务事件消息。

Amazon ECS 服务定义参数

服务定义用于定义如何运行您的 Amazon ECS 服务。可以在服务定义中指定以下参数。

启动类型

`launchType`

类型：字符串

有效值：`EC2` |`FARGATE` |`EXTERNAL`

必需：否

运行您的服务的启动类型。如果没有指定启动类型，将默认使用默认的 `capacityProviderStrategy`。有关更多信息，请参阅 Amazon ECS 启动类型。

如果指定 `launchType`，必须省略 `capacityProviderStrategy` 参数。

容量提供程序策略

`capacityProviderStrategy`

类型： 对象数组

必需：否

要用于服务的容量提供程序策略。

容量提供程序策略由一个或多个容量提供程序以及要分配给它们的 `base` 和 `weight` 组成。容量提供程序必须与要在容量提供程序策略中使用的集群相关联。PutClusterCapacityProviders API 用于将容量提供程序与集群相关联。只能使用具有 `ACTIVE` 或 `UPDATING` 状态的容量提供程序。

如果指定 `capacityProviderStrategy`，必须省略 `launchType` 参数。如果没有指定 `capacityProviderStrategy` 或 `launchType`，将使用集群的 `defaultCapacityProviderStrategy`。

如果要指定使用自动扩缩组的容量提供程序，则必须先创建该容量提供程序。可以通过 CreateCapacityProvider API 操作创建新的容量提供程序。

要使用 AWS Fargate 容量提供程序，请指定 FARGATE 或 FARGATE_SPOT 容量提供程序。AWS Fargate 容量提供程序对所有账户都可用，只需要与要使用的集群相关联。

PutClusterCapacityProviders API 操作用于在创建集群后更新集群的可用容量提供程序列表。

`capacityProvider`

类型：字符串

必需：是

容量提供程序的短名称或完整的 Amazon 资源名称（ARN）。

`weight`

类型：整数

有效范围：介于 0 到 1000 之间的整数。

必需：否

权重值指明使用指定容量提供程序的已启动任务总数的相对百分比。

例如，假定您的策略包含两个容量提供程序，并且两个容量提供程序的权重均为 1。当满足基准时，任务会在两个容量提供程序之间均匀分配。按照相同的逻辑，假定您指定 capacityProviderA 的权重为 1，并指定 capacityProviderB 的权重为 4，那么，对于使用 capacityProviderA 运行的每个任务，都会有四个任务使用 capacityProviderB。

`base`

类型：整数

有效范围：介于 0 到 100000 万之间的整数。

必需：否

基准值指明在指定的容量提供程序上至少运行多少个任务。在一个容量提供程序策略中，只能有一个容量提供程序策略定义了基准。

任务定义

`taskDefinition`

类型：字符串

必需：否

要在您的服务中运行的任务定义的 `family` 和 `revision` (`family:revision`) 或完整 Amazon Resource Name (ARN)。如果未指定 `revision`，则会使用指定系列的最新 `ACTIVE` 版本。

使用滚动更新 (`ECS`) 部署控制器时，必须指定任务定义。

平台操作系统

`platformFamily`

类型：字符串

必需：条件

原定设置：Linux

对于在 Fargate 上托管的 Amazon ECS 服务，此参数是必需的。

对于在 Amazon EC2 上托管的 Amazon ECS 服务，可忽略此参数。

运行服务的容器上的操作系统。有效值为 `LINUX`、`WINDOWS_SERVER_2019_FULL`、`WINDOWS_SERVER_2019_CORE`、`WINDOWS_SERVER_2022_` 和 `WINDOWS_SERVER_2022_CORE`。

为服务指定的每个任务的 `platformFamily` 值都必须与服务 `platformFamily` 值匹配。例如，如果您将 `platformFamily` 设置为 `WINDOWS_SERVER_2019_FULL`，则所有任务的 `platformFamily` 值必须是 `WINDOWS_SERVER_2019_FULL`。

平台版本

`platformVersion`

类型：字符串

必需：否

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 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 适用于 Amazon ECS 的 Fargate Linux 平台版本。

> **Note**
>
> 不会使用 EC2 启动类型为任务指定平台版本。

集群

`cluster`

类型：字符串

必需：否

要在其上运行您的服务的集群的短名称或完整 Amazon Resource Name (ARN)。如果您未指定集群，则采用 `default` 集群。

服务名称

`serviceName`

类型：字符串

必需：是

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。一个集群中的服务名称必须唯一，但是您可以为一个区域或多个区域中多个集群中的服务提供相似的名称。

计划策略

`schedulingStrategy`

类型：字符串

有效值：REPLICA | DAEMON

必需：否

要使用的计划策略。如果未指定计划策略，则使用 REPLICA 策略。有关更多信息，请参阅 Amazon ECS 服务。

有两种服务计划程序策略可用：

- REPLICA：副本计划策略在集群中放置和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。有关更多信息，请参阅 副本策略。
- 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。有关更多信息，请参阅 进程守护程序策略。

> **Note**
>
> Fargate 任务不支持 DAEMON 计划策略。

预期数量

`desiredCount`

类型：整数

必需：否

要在您的服务中放置并保持运行的指定任务定义的实例化数量。

如果使用 REPLICA 计划策略，则需要此参数。如果服务使用 DAEMON 计划策略，则此参数是可选的。

部署配置

`deploymentConfiguration`

类型：对象

必需：否

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

类型：整数

必需：否

如果服务使用滚动更新 (ECS) 部署类型，maximumPercent 参数表示在部署期间允许处于 RUNNING、STOPPING 或 PENDING 状态的服务任务数上限。其表示为向下取整到最近整数的 desiredCount 的百分比。您可以使用此参数定义部署批次大小。例如，如果服务使用 REPLICA 服务调度器且任务的 desiredCount 为 4，maximumPercent 值为 200%，则调度器会在停止 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。使用 REPLICA 服务计划程序的服务的默认 maximumPercent 值为 200%。

如果服务使用的是 DAEMON 服务调度器类型，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 部署类型以及使用 EC2 启动类型的任务，最大百分比值将设置为原定设置值，用于在容器实例处于 RUNNING 状态时，定义服务中保持 DRAINING 状态的任务数上限。如果服务中的任务使用 Fargate 启动类型，则不会使用最大百分比值，但会在描述服务时返回该值。

minimumHealthyPercent

类型：整数

必需：否

如果服务使用滚动更新（ECS）部署类型，minimumHealthyPercent 则表示在部署期间必须处于 RUNNING 状态的服务任务数下限。其表示为向上取整到最近整数的 desiredCount 的百分比。您可以使用此参数进行部署，而无需使用额外的集群容量。例如，如果服务的任务 desiredCount 为 4，minimumHealthyPercent 为 50%，则该服务计划程序可能会在开始 2 个新任务之前停止 2 个现有任务以释放集群容量。

对于不使用负载均衡器的服务，请注意以下事项：

- 如果某个服务中任务内的所有基本容器都通过了运行状况检查，则该服务将被视为正常运行。
- 如果任务没有已定义运行状况检查的基本容器，服务调度器会在任务达到 RUNNING 状态后等待 40 秒，然后将其计入最低正常运行百分比总计。

- 如果任务具有一个或多个已定义运行状况检查的基本容器，服务计划程序会等待任务达到正常运行状态，然后将其计入最低正常运行百分比总计。当某个任务中的所有基本容器都通过了其运行状况检查时，该任务将被视为正常运行。服务计划程序可以等待的时间由容器运行状况检查设置决定。有关更多信息，请参阅 运行状况检查。

对于确实使用负载均衡器的服务，请注意以下事项：

- 如果任务不包含已定义运行状况检查的基本容器，服务计划程序会等待负载均衡器目标组运行状况检查返回正常运行状态，然后将任务计入最低正常运行百分比总计。
- 如果任务具有一个已定义运行状况检查的基本容器，服务计划程序会等待任务达到正常运行状态且负载均衡器目标组运行状况检查返回正常运行状态，然后将任务计入最低正常运行百分比总计。

`minimumHealthyPercent` 的副本服务的默认值为 100%。使用 DAEMON 服务调度的原定设置 `minimumHealthyPercent` 值对于 AWS CLI、AWS 开发工具包和 API 为 0%，对于 AWS Management Console 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` 部署类型以及使用 EC2 启动类型的运行任务，最小正常百分比值将设置为原定设置值，用于在容器实例处于 `RUNNING` 状态时，定义服务中保持在 `DRAINING` 状态的任务数下限。如果服务使用蓝色/绿色 (`CODE_DEPLOY`) 或 `EXTERNAL` 部署类型，并且正在运行使用 Fargate 启动类型的任务，则不会使用最小健康百分比值，尽管在描述服务时会返回该值。

部署控制器

`deploymentController`

类型：对象

必需：否

要用于该服务的部署控制器。如果未指定部署控制器，则使用 `ECS` 控制器。有关更多信息，请参阅 Amazon ECS 服务。

`type`

类型：字符串

有效值：`ECS` |`CODE_DEPLOY` |`EXTERNAL`

必需：是

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。可以调整在服务部署期间允许的最小和最大正常任务数量，以控制滚动更新期间 Amazon ECS 在服务中添加或删除的容器数量，如 deploymentConfiguration 中指定。

CODE_DEPLOY

蓝/绿 (CODE_DEPLOY) 部署类型使用 CodeDeploy 支持的蓝/绿部署模型，以允许您先验证新的服务部署，然后再向其发送生产流量。

EXTERNAL

当您需要使用任何第三方部署控制器来完全控制 Amazon ECS 服务的部署过程时，使用外部部署类型。

任务放置

placementConstraints

类型： 对象数组

必需：否

您的服务中的任务使用的一组放置约束对象。您可以为每个任务指定最多 10 个约束。此限制包括任务定义中的约束以及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

类型：字符串

必需：否

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

类型：字符串

必需：否

应用于约束的集群查询语言表达式。如果约束类型为 distinctInstance，则不能指定表达式。有关更多信息，请参阅 创建表达式，以为 Amazon ECS 任务定义容器实例。

placementStrategy

类型： 对象数组

必需：否

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

类型：字符串

有效值：random |spread |binpack

必需：否

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，则会将任务放置到剩余内存最少但足以运行该任务的实例上。

field

类型：字符串

必需：否

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，例如 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。

标签

tags

类型： 对象数组

必需：否

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。该服务最多可应用 50 个标签。有关更多信息，请参阅 为 Amazon ECS 资源添加标签。

key

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

必需：否

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

类型：字符串

长度约束：最小长度为 0。长度上限为 256。

必需：否

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

类型：布尔值

有效值：true | false

必需：否

指定是否要为该服务内的任务使用 Amazon ECS 托管标签。如果未指定值，则默认值为 false。有关更多信息，请参阅 使用标签记账。

propagateTags

类型：字符串

有效值：TASK_DEFINITION | SERVICE

必需：否

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。要在创建服务或任务以后将标签添加到任务，请使用 TagResource API 操作。

网络配置

`networkConfiguration`

类型：对象

必需：否

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，但其他网络模式不支持该参数。如果使用 Fargate 启动类型，则需要 `awsvpc` 网络模式。有关 Amazon EC2 启动类型联网的更多信息，请参阅EC2 启动类型的 Amazon ECS 任务联网选项。有关 Fargate 启动类型联网的更多信息，请参阅 Fargate 任务联网。

`awsvpcConfiguration`

类型：对象

必需：否

表示任务或服务的子网和安全组的对象。

`subnets`

类型：字符串数组

必需：是

与任务或服务相关联的子网。根据 `awsvpcConfiguration`，可指定的子网数量限制为 16 个。

`securityGroups`

类型：字符串数组

必需：否

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。根据 `awsvpcConfiguration`，可指定的安全组数量限制为 5 个。

`assignPublicIP`

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

任务的弹性网络接口是否接收公有 IP 地址。如果未指定值，则使用默认值 DISABLED。

`healthCheckGracePeriodSeconds`

类型：整数

必需：否

在任务进入 RUNNING 状态后， 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、容器运行状况检查以及 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。如果您的服务定义了负载均衡器，并且您没有指定运行状况检查宽限期值，则使用默认值 0。

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。

如果您不使用 Elastic Load Balancing，我们建议您在任务定义运行状况检查参数中使用 `startPeriod`。有关更多信息，请参阅使用容器运行状况检查确定 Amazon ECS 任务运行状况。

`loadBalancers`

类型： 对象数组

必需：否

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 Application Load Balancer 或 Network Load Balancer 的服务，最多可将 5 个目标组附加到服务。

创建服务后，无法从 AWS Management Console 更改负载均衡器配置。您只能使用 AWS Copilot、AWS CloudFormation、AWS CLI 或 SDK 来修改 ECS 滚动部署控制器的负载均衡器配置，而不是 AWS CodeDeploy 蓝/绿或外部控制器。当您添加、更新或删除负载均衡器配置时，Amazon ECS 会使用更新后的 Elastic Load Balancing 配置启动新部署。这将导致任务注册到负载均衡器或从负载均衡器取消注册。我们建议您在更新 Elastic Load Balancing 配置之前在测试环境中对此进行验证。有关如何修改配置的信息，请参阅 Amazon Elastic Container Service API 参考中的 UpdateService。

对于 Application Load Balancers 和 Network Load Balancers，此对象必须包含负载平衡器目标组 ARN、容器名称（如容器定义中所示）以及要从负载平衡器访问的容器端口。将此服务中的任务放置于容器实例之后，容器实例和端口组合会注册为指定目标组中的目标。

`targetGroupArn`

类型：字符串

必需：否

与服务关联的 Elastic Load Balancing 目标组的完整 Amazon 资源名称（ARN）。

目标组 ARN 仅在使用 Application Load Balancer 或 Network Load Balancer时指定。

`loadBalancerName`

类型：字符串

必需：否

要与该服务关联的负载均衡器的名称。

如果您使用的是 Application Load Balancer 或 Network Load Balancer，则会省略负载均衡器名称参数。

`containerName`

类型：字符串

必需：否

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`

类型：整数

必需：否

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中的任务所使用任务定义中的 `containerPort`。对于使用 EC2 启动类型的任务，容器实例必须允许端口映射的 `hostPort` 上的入站流量。

`role`

类型：字符串

必需：否

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 。仅当为服务使用具有单个目标组的负载平衡器，并且任务定义不使用 awsvpc 网络模式时，才允许使用此参数。如果您指定 role 参数，则还必须指定具有 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 用户指南中的友好名称和路径。

> **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。

`serviceConnectConfiguration`

类型：对象

必需：否

此服务的配置用于发现和连接到服务，以及由命名空间内的其他服务发现和连接。

有关更多信息，请参阅 使用 Service Connect 连接具有短名称的 Amazon ECS 服务。

`enabled`

类型：布尔值

必需：是

指定是否将 Service Connect 与此服务结合使用。

`namespace`

类型：字符串

必需：否

与 Service Connect 结合使用的 AWS Cloud Map 命名空间的 Amazon 资源名称（ARN）短名称或全称。命名空间必须与 Amazon ECS 服务和集群在同一个 AWS 区域 内。命名空间的类型

不影响 Service Connect。有关 AWS Cloud Map 的更多信息，请参阅《AWS Cloud Map 开发人员指南》中的使用服务。

`services`

类型： 对象数组

必需：否

一组 Service Connect 服务对象。这些是其他 Amazon ECS 服务用来连接到此服务的名称和别名（也称为端点）。

对于作为命名空间成员的“客户端”Amazon ECS 服务，只有连接到命名空间内的其他服务时，才需要此字段。例如，接受来自附加到服务的负载均衡器或通过其他方式所传入请求的前端应用程序。

对象从任务定义中选择一个端口，为 AWS Cloud Map 服务分配一个名称，并为客户端应用程序分配别名（也称为端点）数组和端口，以供客户端应用程序引用此服务。

`portName`

类型：字符串

必需：是

`portName` 必须与该 Amazon ECS 服务的任务定义中所有容器中的 `portMappings` 的其中一个的 `name` 相匹配。

`discoveryName`

类型：字符串

必需：否

`discoveryName` 是 Amazon ECS 为该 Amazon ECS 服务创建的新 AWS Cloud Map 服务的名称。此名称在 AWS Cloud Map 命名空间内必须是唯一的。

如果未指定此字段，则使用 `portName`。

`clientAliases`

类型： 对象数组

必需：否

此 Service Connect 服务的客户端别名列表。您可以使用这些列表来分配供客户端应用程序使用的名称。您可以在此列表中包含的最大客户端别名数为 1。

每个别名（“端点”）都是 DNS 名称和端口号，其他 Amazon ECS 服务（“客户端”）可以使用该 DNS 名称和端口号连接到此服务。

每个名称和端口组合在命名空间内必须是唯一的。

这些名称是在客户端服务的每项任务中配置的，而不是在 AWS Cloud Map 中配置的。解析这些名称的 DNS 请求不会离开任务，也不会计入每个弹性网络接口每秒的 DNS 请求的配额。

`port`

类型：整数

必需：是

Service Connect 代理的侦听端口号。此端口在同一个命名空间内的所有任务中都可用。

为避免更改客户端 Amazon ECS 服务中的应用程序，请将其设置为客户端应用程序默认使用的同一个端口。

`dnsName`

类型：字符串

必需：否

`dnsName` 是您在客户端任务的应用程序中用来连接到此服务的名称。该名称必须是有效的 DNS 标签。

如果未指定该字段，则值默认为 `discoveryName.namespace`。如果未指定 `discoveryName`，则使用任务定义中的 `portName`。

为避免更改客户端 Amazon ECS 服务中的应用程序，请将其设置为客户端应用程序默认使用的同一个名称。例如，一些常用名称是 `database`、`db` 或数据库的小写名称，例如 `mysql` 或 `redis`。

`ingressPortOverride`

类型：整数

必需：否

（可选）Service Connect 代理侦听的端口号。

使用此字段的值绕过此应用程序的任务定义中的已命名 `portMapping` 中指定的端口号上的流量代理，然后在 Amazon VPC 安全组中使用该代理，以允许流量进入该 Amazon ECS 服务的代理。

在 `awsvpc` 模式下，默认值是容器端口号，该端口号在此应用程序的任务定义的已命名 `portMapping` 中指定。在 `bridge` 模式下，默认值是 Service Connect 代理的动态临时端口。

`logConfiguration`

类型：LogConfiguration 对象

必需：否

这定义了 Service Connect 代理日志的发布位置。在意外事件期间使用日志进行调试。此配置在此 Amazon ECS 服务的每个任务的 Service Connect 代理容器中设置 `logConfiguration` 参数。没有在任务定义中指定代理容器。

我们建议您使用与该 Amazon ECS 服务任务定义的应用程序容器相同的日志配置。对于 FireLens，这是应用程序容器的日志配置。使用 `fluent-bit` 或 `fluentd` 容器映像的不是 FireLens 日志路由器容器。

`serviceRegistries`

类型： 对象数组

必需：否

服务的服务发现配置的详细信息。有关更多信息，请参阅 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。

`registryArn`

类型：字符串

必需：否

服务注册表的 Amazon 资源名称（ARN）。当前支持的服务注册表是 AWS Cloud Map。有关更多信息，请参阅 AWS Cloud Map 开发人员指南中的使用服务。

`port`

类型：整数

必需：否

在服务发现服务指定 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

类型：字符串

必需：否

用于服务发现服务的容器名称值。此值可在任务定义中指定。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

类型：整数

必需：否

用于服务发现服务的端口值。此值可在任务定义中指定。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。

客户端令牌

clientToken

类型：字符串

必需：否

用于确保请求的幂等性而提供的唯一、区分大小写的标识符。最长可为 32 个 ASCII 字符。

卷配置

volumeConfigurations

类型：对象

必需：否

用于为服务管理的任务而创建卷的配置。为服务中的每项任务创建卷。只有在任务定义中标记为 `configuredAtLaunch` 的卷才能使用此对象进行配置。此对象是将 Amazon EBS 数据卷附加到服务管理的任务所必需的。有关更多信息，请参阅 Amazon EBS 卷。

`name`

类型：字符串

必需：是

创建或更新服务时配置的卷的名称。最多可允许 255 个字母（大写和小写字母）、数字、下划线（_）和连字符（-）。此值必须与任务定义中指定的卷名称相匹配。

`managedEBSVolume`

类型：对象

必需：否

创建或更新服务时附加到服务管理的任务的 Amazon EBS 卷的卷配置。

`encrypted`

类型：布尔值

必需：否

有效值：`true|false`

指定是否对附加到服务管理的任务的 Amazon EBS 卷进行加密。如果您在默认情况下为账户开启了 Amazon EBS 加密，则此设置将被覆盖，且卷将被加密。有关更多信息，请参阅《Amazon EC2 用户指南》中的原定设置加密。

`kmsKeyId`

类型：字符串

必需：否

用于 Amazon EBS 加密的 AWS Key Management Service（AWS KMS）密钥的标识符。如果未指定此参数，则使用 Amazon EBS 的 AWS KMS key。如果指定了 `KmsKeyId`，加密状态必须为 `true`。

您可以使用以下任一项指定 KMS 密钥：

- 密钥 ID – 例如 `1234abcd-12ab-34cd-56ef-1234567890ab`。
- 密钥别名 – 例如 `alias/ExampleAlias`。
- 密钥 ARN – 例如 `arn:aws:kms:us-east-1:012345678910:key/1234abcd-12ab-34cd-56ef-1234567890ab`。
- 别名 ARN – 例如 `arn:aws:kms:us-east-1:012345678910:alias/ExampleAlias`。

> **Important**
>
> AWS 异步验证 KMS 密钥的身份。因此，如果您指定了无效的 ID、别名或 ARN，则操作可能会显示成功，但最终会失败。有关更多信息，请参阅排查 Amazon EBS 卷附加问题。

`volumeType`

类型：字符串

必需：否

有效值：`gp2|gp3|io1|io2|sc1|st1|standard`

EBS 卷的类型。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EBS 卷类型。默认卷类型为 `gp3`。

> **Note**
>
> 配置为附加到 Fargate 任务的 Amazon EBS 卷不支持 `standard` 卷类型。

`sizeInGiB`

类型：整数

必需：否

有效范围：介于 1 到 16384 之间的整数

EBS 卷的大小，以吉字节（GiB）为单位。如果您没有提供快照 ID 来配置要附加的卷，则必须提供大小值。如果使用快照配置卷以进行附加，则默认值为快照大小。然后，您可以指定大于或等于快照大小的大小。

对于 gp2 和 gp3 卷类型，有效范围为 1-16384。

对于 io1 和 io2 卷类型，有效范围为 4-16384。

对于 st1 和 sc1 卷类型，有效范围为 125-16384。

对于 standard 卷类型，有效范围为 1-1024。

snapshotId

类型：字符串

必需：否

用于创建附加到 ECS 任务的新卷的现有 EBS 卷的快照 ID。

iops

类型：整数

必需：否

每秒 I/O 操作数 (IOPS)。对于 gp3、io1 和 io2 卷，这代表为卷预置的 IOPS 数。对于 gp2 卷，此值表示卷的基准性能和卷积累用于突增情况的 I/O 积分的速度。此参数是 io1 和 io2 卷的必需参数。gp2、st1、sc1 或 standard 卷不支持此参数。

对于 gp3 卷，值的有效范围为 3000 到 16000。

对于 io1 卷，值的有效范围为 100 到 64000。

对于 io2 卷，值的有效范围为 100 到 64000。

throughput

类型：整数

必需：否

为附加到服务管理的任务的卷预调配的吞吐量。

> **Important**
>
> 仅 gp3 卷支持此参数。

`roleArn`

类型：字符串

必需：是

基础设施 AWS Identity and Access Management（IAM）角色的 Amazon 资源 ARN（ARN），它为您的任务提供 Amazon ECS 管理 Amazon EBS 资源的权限。有关更多信息，请参阅 Amazon ECS 基础设施 IAM 角色。

`tagSpecifications`

类型：对象

必需：否

应用于 Amazon EBS 卷管理的服务的标签规范。

`resourceType`

类型：字符串

必需：是

有效值：`volume`

要在创建时标记的资源类型。

`tags`

类型： 对象数组

必需：否

您应用于卷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。`AmazonECSCreated` 和 `AmazonECSManaged` 是 Amazon ECS 代表您添加的保留标签，因此您最多可以自己指定 48 个标签。在卷被删除时，标签也会随之被删除。有关更多信息，请参阅 为 Amazon ECS 资源添加标签。

key

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

必需：否

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

类型：字符串

长度约束：最小长度为 0。长度上限为 256。

必需：否

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。

propagateTags

类型：字符串

有效值：TASK_DEFINITION |SERVICE |NONE

必需：否

指定是否要将标签从任务定义或服务复制到卷中。如果指定 NONE 或未指定任何值，则不会复制标签。

fileSystemType

类型：字符串

必需：否

有效值：xfs|ext3|ext4

卷上的文件系统的类型。卷的文件系统类型会决定在卷中存储和检索数据的方式。对于从快照创建的卷，必须指定创建快照时卷使用的相同文件系统类型。如果文件系统类型不匹配，则任务将无法启动。附加到 Linux 任务的卷的默认值为 XFS。

服务定义模板

下面显示了 Amazon ECS 服务定义的 JSON 表示形式。

Amazon EC2 启动类型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
```

```
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
```

```
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
```

```
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Fargate 启动类型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "FARGATE",
    "capacityProviderStrategy": [
```

```
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "platformFamily": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
```

```
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
```

```
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

您可以使用以下 AWS CLI 命令创建此服务定义模板。

```
aws ecs create-service --generate-cli-skeleton
```

为 Amazon ECS 资源添加标签

为了帮助您管理 Amazon ECS 资源，您可以选择使用标签将自己的元数据分配给任何基于 ARN 的资源。每个标签都由一个键和一个可选值组成。

您可以使用标签按照不同方式（例如按用途、所有者或环境）对 Amazon ECS 资源进行分类。这在您有许多相同类型的资源时会非常有用。您可以根据分配到特定资源的标签来快速识别该资源。例如，您可以为您账户的 Amazon ECS 实例定义一组标签。这有助于您跟踪每个实例的所有者和堆栈级别。

您可以将标签用于成本和使用情况报告。您可以使用这些报告来分析 Amazon ECS 资源的成本和使用情况。有关更多信息，请参阅 the section called “使用情况报告”。

> **Warning**
>
> 许多 API 会返回标签密钥及其值。拒绝访问 `DescribeTags` 不会自动拒绝访问其他 API 返回的标签。作为最佳实践，我们建议您不要在标签中包含敏感数据。

我们建议您针对每类资源设计一组标签，以满足您的需要。您可以使用一组一致的标签键来更轻松地管理您的资源。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。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键与该资源上现有标签的键相同，则新值会覆盖旧值。删除资源时，资源的所有标签也会被删除。

如果您使用的是 AWS Identity and Access Management（IAM），则可以控制您的AWS账户中的哪些用户拥有管理标签的权限。

如何为资源添加标签

有多种方式可以为 Amazon ECS 任务、服务、任务定义和集群添加标签：

- 用户使用 AWS Management Console、Amazon ECS API、AWS CLI 或 AWS SDK 为资源手动添加标签。
- 用户创建服务或运行独立任务，然后选择 Amazon ECS 托管标签选项。

  Amazon ECS 会自动为所有新启动的任务添加标签。有关更多信息，请参阅 the section called “Amazon ECS 托管标签”。

- 用户使用控制台创建资源。控制台会自动为资源添加标签。

  这些标签在 AWS CLI 和 AWS SDK 响应中返回，并显示在控制台中。您无法修改或删除这些标签。

  有关添加的标签的信息，请参阅 Amazon ECS 资源标签支持表中的控制台自动添加的标签列。

如果您在创建资源时指定了标签但无法应用标签，Amazon ECS 会回滚创建过程。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。通过在创建资源时对其进行标记，无需在创建资源后运行自定义标记脚本。

下表描述了支持标签的 Amazon ECS 资源。

Amazon ECS 资源标记支持

资源	支持标签	支持标签传播	控制台自动添加的标签
Amazon ECS 任务	是	是，从任务定义。	键：`aws:ecs:clusterName` 值：`cluster-name`
Amazon ECS 服务	是	是，从任务定义或服务到服务中的任务。	键：`ecs:service:stackId` 值 `arn:aws:cloudformation: arn`
Amazon ECS 任务集	是	不支持	不适用
Amazon ECS 任务定义	是	不支持	键：`ecs:taskDefinition:createdFrom` 值：`ecs-console-v2`

资源	支持标签	支持标签传播	控制台自动添加的标签
Amazon ECS 集群	是	不支持	键：aws:cloudformation:logical-id 值：ECSCluster 键：aws:cloudformation:stack-id 值：arn:aws:cloudformation: *arn* 键：aws:cloudformation:stack-name 值：ECS-Console-V2-Cluster- *EXAMPLE*
Amazon ECS 容器实例。	是	是，从 Amazon EC2 实例。有关更多信息，请参阅 向 Amazon ECS 容器实例添加标签。	不适用
Amazon ECS 外部实例	是	不支持	不适用

资源	支持标签	支持标签传播	控制台自动添加的标签
Amazon ECS 容量提供程序	是。 您不能为预定义的 FARGATE 和 FARGATE_SPOT 容量提供程序添加标签。	否	不适用

在创建时标记资源

以下资源支持在创建时使用 Amazon ECS API、AWS CLI 或 AWS SDK 进行标记：

- Amazon ECS 任务
- Amazon ECS 服务
- Amazon ECS 任务定义
- Amazon ECS 任务集
- Amazon ECS 集群
- Amazon ECS 容器实例。
- Amazon ECS 容量提供程序

Amazon ECS 可以选择使用标记授权来创建资源。配置 AWS 账户 进行标记授权时，用户必须对创建资源的操作具有权限，例如 `ecsCreateCluster`。在资源创建操作中指定标签时，AWS 会执行额外的授权，以验证用户或角色是否具备创建标签的权限。因此，您必须授予使用 `ecs:TagResource` 操作的显式权限。有关更多信息，请参阅 the section called “在创建过程中，为资源添加标签”。有关如何配置选项的信息，请参阅the section called “标记授权”。

限制

以下限制适用于标签：

- 一个资源最多可以关联 50 个标签。
- 不能对一个资源重复使用标签键。每个标签键必须具有唯一性，而且只能有一个值。

- 键最长可达 128 个字符（采用 UTF-8 格式）。
- 值最长可达 256 个字符（采用 UTF-8 格式）。
- 如果有多个 AWS 服务 和资源使用您的标记方案，请限制您使用的字符类型。某些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字母、数字、空格以及以下字符：**+ - = . _ : / @**。
- 标签键和值区分大小写。
- 不能使用 `aws:`、`AWS:` 或它们的任何大写或小写组合作为键或值的前缀。它们保留供 AWS 使用。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。

Amazon ECS 托管标签

当您使用 Amazon ECS 托管标签时，Amazon ECS 会自动使用集群信息和用户添加的任务定义标签或服务标签来标记所有新启动的任务和附加到任务的任何 Amazon EBS 卷。下面介绍了添加的标签：

- 独立任务 – 键为 `aws:ecs:clusterName` 且值设置为集群名称的标签。用户添加的所有任务定义标签。附加到独立任务的 Amazon EBS 卷将收到带有密钥为 `aws:ecs:clusterName`、值设置为集群名称的标签。有关 Amazon EBS 卷标记的更多信息，请参阅标记 Amazon EBS 卷。
- 服务所包含的任务 – 键为 `aws:ecs:clusterName` 且值设置为集群名称的标签。键为 `aws:ecs:serviceName` 且值设置为服务名称的标签。来自以下资源之一的标签：
  - 任务定义 – 用户添加的所有任务定义标签。
  - 服务 – 用户添加的所有服务标签。

    附加到作为服务一部分的任务的 Amazon EBS 卷将收到一个密钥为 `aws:ecs:clusterName`、值设置为集群名称的标签，以及一个密钥为 `aws:ecs:serviceName`、值设置为服务名称的标签。有关 Amazon EBS 卷标记的更多信息，请参阅标记 Amazon EBS 卷。

此功能需要以下选项：

- 您必须选择使用新的 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 Resource Names (ARN) 和 ID。
- 当您使用 API 创建服务或运行任务时，您必须将 `run-task` 和 `create-service` 的 `enableECSManagedTags` 设置为 `true`。有关更多信息，请参阅 AWS Command Line Interface API 参考中的 create-service 和 run-task。
- Amazon ECS 使用托管标签来确定何时启用某些功能，例如集群自动扩缩。建议您不要手动修改标签，以便 Amazon ECS 可以有效地管理这些功能。

使用标签记账

AWS 提供了名为 Cost Explorer 的报告工具，您可以使用它来分析 Amazon ECS 资源的成本和用量。

您可以使用 Cost Explorer 查看用量和成本的图表。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 查看您在 AWS 资源上花费的时间模式。如，您可以使用它来确定需要进一步查询的方面，并查看可用于了解成本的趋势。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。

您可以为成本和使用情况报告使用 Amazon ECS 托管标签或用户添加的标签。有关更多信息，请参阅 Amazon ECS 使用率报告。

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。有关设置带有标签的成本分配报告的更多信息，请参阅 AWS Billing 用户指南中的月度成本分配报告。

此外，您可以启用拆分成本分配数据，以获取成本和使用量报告中的任务级 CPU 和内存使用情况数据。有关更多信息，请参阅 任务级成本和使用量报告。

> **Note**
>
> 如果已启用报告，则最多需要 24 小时才能后查看当月的数据。

向 Amazon ECS 资源添加标签

您可以标记新的或现有的任务、服务、任务定义或集群。有关标记容器实例的信息，请参阅 向 Amazon ECS 容器实例添加标签。

> **Warning**
>
> 请勿在标签中添加个人身份信息（PII）或其他机密或敏感信息。标签可供许多 AWS 服务访问，包括计费。标签不适合用于私有或敏感数据。

在创建资源时，您可以使用以下资源指定标签。

任务	控制台	AWS CLI	API 操作
运行一个或多个任务。	将应用程序作为 Amazon ECS 任务运行	run-task	RunTask
创建服务。	使用控制台创建 Amazon ECS 服务	create-service	CreateService
创建任务集。	使用第三方控制器部署 Amazon ECS 服务	create-task-set	CreateTaskSet
注册任务定义。	the section called “使用控制台创建任务定义”	register-task-definition	RegisterTaskDefinition
创建集群。	创建 Fargate 启动类型的 Amazon ECS 集群	create-cluster	CreateCluster
运行一个或多个容器实例。	启动 Amazon ECS Linux 容器实例	run-instances	RunInstances

为现有资源添加标签（Amazon ECS 控制台）

您可以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。

为单个资源修改标签

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 从导航栏中，选择要使用的AWS 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，选择 Tags（标签）选项卡，然后选择 Manage tags（管理标签）。

5. 配置您的标签。

   [添加标签] 选择 Add tag（添加标签），然后执行以下操作：

   - 对于 Key（键），输入键名称。
   - 对于值，输入键值。

6. 选择保存。

向现有资源添加标签（AWS CLI）

您可以使用 AWS CLI 或 API 添加或覆盖一个或多个标签。

- AWS CLI - tag-resource
- API 操作 - TagResource

向 Amazon ECS 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 引导启动 Amazon ECS Linux 容器实例以传递数据。

  > **Important**
  >
  > 如果使用 Amazon EC2 Auto Scaling 组启动容器实例，则应使用 ECS_CONTAINER_INSTANCE_TAGS 配置参数添加标签。这是由标签添加到使用 自动扩缩组启动的 Amazon EC2 实例的方式造成的。

  下面是将标签与您的容器实例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 – 当您使用 Amazon EC2 API、CLI 或控制台创建容器实例时，请首先使用 `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` 将用户数据传递给实例。这样做会将它们从 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 policy。

  - `ec2:DescribeTags`

  以下是用于添加这些权限的示例策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
             "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
    ]
}
```

外部容器实例

您可以使用以下方法之一将标签与您的外部容器实例相关联。

- 方法 1–在运行安装脚本向集群注册外部实例之前，请在 `/etc/ecs/ecs.config` 处创建或编辑 Amazon ECS 容器代理配置文件，并添加 `ECS_CONTAINER_INSTANCE_TAGS` 容器代理配置参数。这将创建与外部实例关联的标签。

  以下是示例语法。

  ```
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  ```

- 方法 2 – 将外部实例注册到集群后，您可以使用 AWS Management Console 添加标签。有关更多信息，请参阅 为现有资源添加标签（Amazon ECS 控制台）。

Amazon ECS 使用率报告

AWS 提供了名为 Cost Explorer 的报告工具，您可以使用它来分析 Amazon ECS 资源的成本和用量。

您可以使用 Cost Explorer 查看用量和成本的图表。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 查看您在 AWS 资源上花费的时间模式。如，您可以使用它来确定需要进一步查询的方面，并查看可用于了解成本的趋势。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。

成本和使用情况报告中的计量数据显示了所有 Amazon ECS 任务的使用情况。计量数据包括运行的每个任务的 CPU 用量（用 `vCPU-Hours` 表示）和内存用量（用 `GB-Hours` 表示）。数据的显示方式取决于任务的启动类型。

对于使用 Fargate 启动类型的任务，`lineItem/Operation` 列将显示 `FargateTask`，并且您将看到与每个任务关联的成本。

对于使用 EC2 启动类型的任务，`lineItem/Operation` 列会显示 `ECSTask-EC2`，并且任务没有关联的直接成本。报告中显示的计量数据（例如内存使用情况）表示任务在您指定的计费周期内保留的总资源。您可以使用此数据来确定 Amazon EC2 实例的基础集群的成本。您的 Amazon EC2 实例的成本和用量数据将在 Amazon EC2 服务下单独列出。

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 使用标签记账。

> **Important**
>
> 只能查看在 2018 年 11 月 16 日或之后启动的任务的计量数据。此日期之前启动的任务不显示计量数据。

下面是可用于在 Cost Explorer 中对成本分配数据进行排序的一些字段的示例。

- 集群名称
- 服务名称
- 资源标签
- 启动类型
- AWS 区域
- 使用情况类型

有关创建 AWS 成本和用量报告的更多信息，请参阅 AWS Billing 用户指南中的AWS成本和用量报告。

任务级成本和使用量报告

AWS Cost Management 可以在 AWS 成本和使用情况报告 中为 Amazon ECS 上的每项任务（包括 Fargate 上的任务和 EC2 上的任务）提供 CPU 和内存使用量数据。此数据称为拆分成本分配数据。您可以使用这些数据来分析应用程序的成本和使用量。此外，您还可以使用成本分配标签和成本类别将成本拆分并分配给各个业务部门和团队。有关拆分成本分配数据的更多信息，请参阅《AWS 成本和使用情况报告 用户指南》中的了解拆分成本分配数据。

您可以在 AWS Cost Management Console 中为账户选择任务级拆分成本分配数据。如果您有管理（付款人）账户，则可以选择加入付款人账户，以将此配置应用于每个关联账户。

设置拆分成本分配数据后，报告中的 splitLineItem 标题下将有其他的列。有关更多信息，请参阅《AWS 成本和使用情况报告 用户指南》中的拆分行项目详细信息

对于 EC2 上的任务，此数据根据资源使用量或预留量和实例上的剩余资源来分配 EC2 实例的成本。

以下是先决条件：

- 将 `ECS_DISABLE_METRICS` Amazon ECS 代理配置参数设置为 `false`。

  此设置为 `false` 时，Amazon ECS 代理将指标发送到 Amazon CloudWatch。在 Linux 上，此设置默认为 `false`，且指标将发送到 CloudWatch。在 Windows 上，此设置默认为 `true`，因此您必须将设置更改为 `false` 才能将指标发送到 CloudWatch 以供 AWS Cost Management 使用。有关 ECS 代理配置的更多信息，请参阅 Amazon ECS 容器代理配置。
- 可靠指标的最低 Docker 版本是 Docker 版本 v20.10.13 及更高版本，该版本包含在经 Amazon ECS 优化的 AMI 20220607 及更高版本中。

要使用拆分成本分配数据，您必须创建报告，然后选择拆分成本分配数据。有关更多信息，请参阅《AWS 成本和使用情况报告 用户指南》中的创建成本和使用量报告。

AWS Cost Management 使用任务 CPU 和内存使用量计算拆分成本分配数据。如果使用量不可用，则 AWS Cost Management 可以使用任务 CPU 和内存预留来代替使用量。如果您看到 CUR 正在使用预留，请检查您的容器实例是否满足先决条件以及任务资源使用量指标是否显示在 CloudWatch 中。

监控 Amazon ECS

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。在开始监控 Amazon ECS 之前，创建一个其中包括以下问题答案的监控计划：

- 监控目的是什么？
- 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视您的服务。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，至少应监控以下项目：

- 您的 Amazon ECS 集群的 CPU 以及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

  有关更多信息，请参阅 查看 Amazon ECS 指标。

监控 Amazon ECS 的最佳实践

使用监控 Amazon ECS 的以下最佳实践。

- 让监控成为优先事务，阻止小问题演变为大问题
- 创建一个监控计划，其中包括以下问题的答案
  - 监控目的是什么？
  - 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？
- 尽可能自动执行监控。
- 检查 Amazon ECS 日志文件。有关更多信息，请参阅 查看 Amazon ECS 容器代理日志。

Amazon ECS 的监控工具

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，并在出现错误时进行报告：

- Amazon CloudWatch 告警 – 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。具体操作是：通知已发送到 Amazon Simple Notification Service（Amazon SNS）主题或 Amazon EC2 Auto Scaling 策略。CloudWatch 告警不调用操作，因为这些操作处于特定状态；状态必须改变并保持指定时间。有关更多信息，请参阅 使用 CloudWatch 监控 Amazon ECS 。

  对于具有使用 Fargate 启动类型的任务的服务，您可以使用 CloudWatch 警报根据 CloudWatch 指标（如 CPU 和内存利用率）在服务中扩展或缩减任务。有关更多信息，请参阅 自动扩展 Amazon ECS 服务。

  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）横向缩减和横向扩展容器实例。

  对于使用经 Amazon ECS 优化的 Amazon Linux AMI 启动的容器实例，您可以使用 CloudWatch Logs 在一个方便的位置查看容器实例中的不同日志。您必须在容器实例上安装 CloudWatch 代理。有关更多信息，请参阅 Amazon CloudWatch用户指南中的使用命令行下载并配置 CloudWatch 代理。您还必须将 `ECS-CloudWatchLogs` 策略添加到 `ecsInstanceRole` 角色。有关更多信息，请参阅 监控容器实例的权限。
- 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。有关更多信息，请参阅 将 Amazon ECS 日志发送到 CloudWatch 。

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。

- 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，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 使用 EventBridge 自动响应 Amazon ECS 错误 以及 Amazon CloudWatch Events 用户指南中的什么是 Amazon CloudWatch Events？
- 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。Container Insights 使用嵌入式指标格式将数据收集为性能日志事件。这些性能日志事件是使用结构化 JSON 架构的条目，允许批量提取和存储高基数数据。从该数据中，CloudWatch 在集群、任务和服务级别创建聚合指标作为 CloudWatch 指标。Container Insights 收集的指标可在 CloudWatch 自动控制面板中使用，也可在 CloudWatch 控制台的指标部分查看。
- 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 来进行实时监控，用 Java 编写日志处理应用程序，验证 CloudTrail 提供的日志文件未发生更改。有关更多信息，请参阅本指南中的 使用 AWS CloudTrail 记录 Amazon ECS API 调用 和 AWS CloudTrail用户指南中的使用 CloudTrail 日志文件。
- 运行时监控 – 检测 AWS 环境中集群和容器的威胁。运行时监控使用 GuardDuty 安全代理为单个 Amazon ECS 工作负载增加运行时可见性，例如，文件访问、进程执行和网络连接。

手动监控工具

监控 Amazon ECS 的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。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。

- Amazon ECS 控制台：
  - EC2 启动类型的集群指标
  - 服务指标
  - 服务运行状况
  - 服务部署事件
- CloudWatch 主页：
  - 当前告警和状态
  - 告警和资源图表
  - 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。
- 容器运行状况检查 - 这些命令在容器上本地运行，可验证应用程序的运行状况和可用性。您可以在任务定义中为每个容器配置这些命令。
- 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。

  Trusted Advisor 具有与 Amazon ECS 相关的以下检查：
  - 一种容错能力，表示您有一项服务在单个可用区中运行。
  - 一种容错能力，表示您未对多个可用区使用分散放置策略。
- AWS Compute Optimizer 是一种服务，用于分析 AWS 资源的配置和利用率指标。它将报告您的资源是否处于最佳状态并生成优化建议，以降低成本并提高工作负载的性能。

  有关更多信息，请参阅 针对 Amazon ECS 的 AWS Compute Optimizer 建议。

使用 CloudWatch 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，以便您能够访问历史信息并更好地了解您的群集或服务的运行情况。预设情况下，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。

Amazon ECS 为集群和服务提供免费的指标。对于附加成本，您可以为集群启用 Amazon ECS CloudWatch Container Insights，了解每个任务的指标，包括 CPU、内存和 EBS 文件系统利用率。有关安装 Container Insights 的更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。

注意事项

使用 Amazon ECS CloudWatch 指标时应考虑以下内容。

- Fargate 上托管的任何 Amazon ECS 服务都会自动启用 CloudWatch CPU 和内存利用率指标，因此您无需采取任何手动步骤。

- 对于在 Amazon EC2 实例上托管的任何 Amazon ECS 任务或服务，Amazon EC2 实例都需要版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0` 或更高版本（Windows）的容器代理才能生成 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。
- 为了获得可靠的 CloudWatch 指标，最低的 Docker 版本是 Docker `20.10.13` 版本及更高版本。
- 您的 Amazon ECS 容器实例还需要对您用来启动 Amazon EC2 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 IAM 角色，则可能需要添加此权限。有关检查容器实例 IAM 角色和为容器实例附加托管 IAM 策略的信息，请参阅Amazon ECS 容器实例 IAM 角色。
- 您可以通过在 Amazon ECS 容器代理配置中设置 `ECS_DISABLE_METRICS=true` 来禁用 Amazon EC2 实例上的 CloudWatch 指标收集。有关更多信息，请参阅 Amazon ECS 容器代理配置。

推荐的指标

Amazon ECS 为您提供用来监控资源的免费 CloudWatch 指标。整个集群中的 CPU 和内存预留以及 CPU、内存和 EBS 文件系统利用率，以及集群中的服务上的 CPU、内存和 EBS 文件系统利用率，都可以使用这些指标进行测量。对于您的 GPU 工作负载，您可以度量集群中的 GPU 预留。

在集群中托管 Amazon ECS 任务的基础设施决定了哪些指标可用。对于在 Fargate 基础设施上托管的任务，Amazon ECS 提供 CPU、内存，并提供 EBS 文件系统利用率指标来帮助监控您的服务。对于在 EC2 实例上托管的任务，Amazon ECS 在集群和服务级别上提供 CPU、内存和 GPU 预留指标以及 CPU 和内存利用率指标。您需要单独监控构成底层基础设施的 Amazon EC2 实例。有关监控 Amazon EC2 实例的更多信息，请参阅《Amazon EC2 用户指南》中的监控 Amazon EC2。

有关与 Amazon ECS 一起使用的建议警报的信息，请参阅《Amazon CloudWatch Logs 用户指南》中的以下内容之一：

- Amazon ECS
- 具有 Container Insights 的 Amazon ECS

查看 Amazon ECS 指标

在集群中运行资源后，您可以在 Amazon ECS 和 CloudWatch 控制台上查看指标。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。

Amazon ECS 控制台

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 Amazon ECS 服务利用率指标。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 选择要查看其指标的集群。
3. 请确定要查看的指标。

要查看指标来源	步骤
集群	在“集群详细信息”页面上，选择指标选项卡。如果您开启了 CloudWatch Container Insights 指标，则还会提供一个指向 CloudWatch 控制台的链接，用于查看这些指标。
服务	在“集群详细信息”页面的服务选项卡上，选择服务。然后，这些指标显示在运行状况和指标选项卡中。

CloudWatch 控制台

对于 Fargate 启动类型，也可在 CloudWatch 控制台上查看 Amazon ECS 服务指标。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。您可以查看服务利用率和正在运行服务的任务计数。

对于 EC2 启动类型，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。

有关如何查看指标的更多信息，请参阅《Amazon CloudWatch 用户指南》中的查看可用指标。

Amazon ECS CloudWatch 指标

您可以使用 CloudWatch 用量指标来提供账户资源使用情况的可见性。这些指标可在 CloudWatch 图表和控制面板上直观呈现当前的服务使用情况。

`CPUReservation`

集群或服务中保留的 CPU 单位的百分比。

CPU 预留（按 `ClusterName` 进行筛选）是用集群上 Amazon ECS 任务预留的总 CPU 单位数除以集群中注册的所有 Amazon EC2 实例的总 CPU 单位数计算得到的。仅处于 `ACTIVE` 或 `DRAINING` 状态的 Amazon EC2 实例将影响 CPU 预留指标。此指标仅支持 Amazon EC2 实例上托管的任务。

有效维度：`ClusterName`。

有用的统计数据：Average、Minimum、Maximum

单位：百分比。

`CPUUtilization`

集群或服务中使用的 CPU 单位的百分比。

集群级别的 CPU 使用率（按 `ClusterName` 筛选）是通过将集群上的 Amazon ECS 任务所使用的总 CPU 单位数除以集群中注册的所有 Amazon EC2 实例的总 CPU 单位数计算得到的。仅处于 `ACTIVE` 或 `DRAINING` 状态的 Amazon EC2 实例将影响 CPU 预留指标。集群级别的指标仅支持 Amazon EC2 实例上托管的任务。

服务级别的 CPU 使用率（按 `ClusterName`、`ServiceName` 进行筛选）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。服务级别的指标仅支持 Amazon EC2 实例和 Fargate 上托管的任务。

有效维度：`ClusterName`、`ServiceName`。

有用的统计数据：Average、Minimum、Maximum

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。

集群内存预留是通过将集群上的 Amazon ECS 任务在集群上预留的总内存量除以集群中注册的所有 Amazon EC2 实例的总内存量计算得到的。此指标只能按 ClusterName 筛选。仅处于 ACTIVE 或 DRAINING 状态的 Amazon EC2 实例将影响内存预留指标。集群级别的内存预留指标仅支持 Amazon EC2 实例上托管的任务。

> **Note**
>
> 计算内存利用率时，如果 MemoryReservation 已指定，则在计算中使用它而不是总内存。

有效维度：ClusterName。

有用的统计数据：Average、Minimum、Maximum

单位：百分比。

MemoryUtilization

集群或服务所使用的内存的百分比。

集群级别的内存使用率（按 ClusterName 筛选）是通过将集群上的 Amazon ECS 任务所使用的总内存量除以集群中注册的所有 Amazon EC2 实例的总内存量计算得到的。仅处于 ACTIVE 或 DRAINING 状态的 Amazon EC2 实例将影响内存利用率指标。集群级别的指标仅支持 Amazon EC2 实例上托管的任务。

服务级别的内存使用率（按 ClusterName、ServiceName 进行筛选）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。服务级别的指标仅支持 Amazon EC2 实例和 Fargate 上托管的任务。

有效维度：ClusterName、ServiceName。

有用的统计数据：Average、Minimum、Maximum

单位：百分比。

EBSFilesystemUtilization

服务中的任务使用的 Amazon EBS 文件系统的百分比。

服务级别的 EBS 文件系统利用率指标（按 ClusterName、ServiceName 筛选）是通过将属于该服务的任务使用的 EBS 文件系统的总量，除以为属于该服务的所有任务分配的 EBS 文件系统

存储总量来计算得到的。服务级别的 EBS 文件系统利用率指标仅适用于托管在附加了 EBS 卷的 Amazon EC2 实例（使用容器代理版本 `1.79.0`）和 Fargate（使用平台版本 `1.4.0`）上的任务。

> **Note**
>
> 对于托管在 Fargate 上的任务，磁盘上有仅供 Fargate 使用的空间。Fargate 使用的空间不会产生任何成本，但可以使用类似 `df` 的工具看到额外的存储空间。

有效维度：`ClusterName, ServiceName`。

有用的统计数据：Average、Minimum、Maximum

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。

集群级别的 GPU 预留指标是通过将是集群上 Amazon ECS 任务预留的 GPU 数除以集群中注册的所有具有 GPU 的容器实例上可用的 GPU 总数计算得出的。仅处于 `ACTIVE` 或 `DRAINING` 状态的 Amazon EC2 实例将影响 GPU 预留指标。

有效维度：`ClusterName`。

有用的统计数据：Average、Minimum、Maximum

所有统计数据：平均值、最小值、最大值、总计、样本数。

单位：百分比。

`ActiveConnectionCount`

从客户端到在共享选定 `DiscoveryName` 的任务中运行的 Amazon ECS Service Connect 代理的并发活动连接总数。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`NewConnectionCount`

从客户端到在共享选定 `DiscoveryName` 的任务中运行的 Amazon ECS Service Connect 代理建立的新连接总数。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`ProcessedBytes`

Service Connect 代理处理的入站流量的总字节数。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：字节。

`RequestCount`

Service Connect 代理处理的入站流量请求的数量。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

您还需要在任务定义的端口映射中配置 `appProtocol`。

有效维度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`GrpcRequestCount`

Service Connect 代理处理的 gRPC 入站流量请求的数量。

此指标仅在您配置了 Amazon ECS Service Connect 且 `appProtocol` 为位于任务定义的端口映射中的 `GRPC` 时才可用。

有效维度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`HTTPCode_Target_2XX_Count`

应用程序在这些任务中生成的编号为 200 到 299 的 HTTP 响应代码的数量。这些任务是目标。此指标仅计入应用程序在这些任务中发送给 Service Connect 代理的响应，不计入直接发送的响应。

此指标仅在您配置了 Amazon ECS Service Connect 且 `appProtocol` 为位于任务定义的端口映射中的 HTTP 或 HTTP2 时才可用。

有效维度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`HTTPCode_Target_3XX_Count`

应用程序在这些任务中生成的编号为 300 到 399 的 HTTP 响应代码的数量。这些任务是目标。此指标仅计入应用程序在这些任务中发送给 Service Connect 代理的响应，不计入直接发送的响应。

此指标仅在您配置了 Amazon ECS Service Connect 且 `appProtocol` 为位于任务定义的端口映射中的 HTTP 或 HTTP2 时才可用。

有效维度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`HTTPCode_Target_4XX_Count`

应用程序在这些任务中生成的编号为 400 到 499 的 HTTP 响应代码的数量。这些任务是目标。此指标仅计入应用程序在这些任务中发送给 Service Connect 代理的响应，不计入直接发送的响应。

此指标仅在您配置了 Amazon ECS Service Connect 且 `appProtocol` 为位于任务定义的端口映射中的 HTTP 或 HTTP2 时才可用。

有效维度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计

单位：计数。

`HTTPCode_Target_5XX_Count`

应用程序在这些任务中生成的编号为 500 到 599 的 HTTP 响应代码的数量。这些任务是目标。此指标仅计入应用程序在这些任务中发送给 Service Connect 代理的响应，不计入直接发送的响应。

此指标仅在您配置了 Amazon ECS Service Connect 且 `appProtocol` 为位于任务定义的端口映射中的 `HTTP` 或 `HTTP2` 时才可用。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

`RequestCountPerTarget`

共享所选 `DiscoveryName` 的每个目标收到的平均请求数。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值。

单位：计数。

`TargetProcessedBytes`

Service Connect 代理处理的总字节数。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

有用的统计数据：平均值、最小值、最大值、总计。

单位：字节。

`TargetResponseTime`

应用程序请求处理的延迟。请求到达目标任务中的 Service Connect 代理到从目标应用程序收到返回代理的响应所用的时间（以毫秒为单位）。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName。

有用的统计数据：平均值、最小值、最大值。

所有统计数据：平均值、最小值、最大值、总计、样本数。

单位：毫秒。

ClientTLSNegotiationErrorCount

TLS 连接失败的总次数。此指标仅在启用 TLS 时使用。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：DiscoveryName 和 DiscoveryName、ServiceName、ClusterName。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

TargetTLSNegotiationErrorCount

由于缺少客户端证书、AWS Private CA 验证失败或 SAN 验证失败而导致 TLS 连接失败的总次数。此指标仅在启用 TLS 时使用。

此指标仅在您配置了 Amazon ECS Service Connect 时可用。

有效维度：ServiceName、ClusterName、TargetDiscoveryName 和 TargetDiscoveryName。

有用的统计数据：平均值、最小值、最大值、总计。

单位：计数。

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。Amazon ECS 仅为任务处于 RUNNING 状态的资源发送指标。例如，如果您的集群包含一个服务，但该服务没有处于 RUNNING 状态的任务，则不会向 CloudWatch 发送任何指标。如果您有两个服务，其中一个服务具有正在运行的任务，而另一个服务没有，则仅发送具有正在运行的任务的服务的指标。

`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。

`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。

`DiscoveryName`

该维度将您为流量指标请求的数据筛选到所有 Amazon ECS 集群的指定 Service Connect 发现名称。

请注意，正在运行的容器中的特定端口可能有多个发现名称。

`DiscoveryName, ServiceName, ClusterName`

此维度会将您请求的流量指标数据筛选到所有任务中的指定 Service Connect 发现名称，这些任务具有此发现名称且由此服务在此集群中创建。

如果您在不同命名空间的多个服务中重复使用了相同的发现名称，则可以使用此维度来查看特定服务的入站流量指标。

请注意，正在运行的容器中的特定端口可能有多个发现名称。

`TargetDiscoveryName`

该维度将您为流量指标请求的数据筛选到所有 Amazon ECS 集群的指定 Service Connect 发现名称。

与 `DiscoveryName` 不同的是，这些流量指标仅衡量来自在此命名空间中具有 Service Connect 配置的其他 Amazon ECS 任务的传送到该 `DiscoveryName` 的入站流量。这包括具有仅限客户端配置或客户端/服务器 Service Connect 配置的服务执行的任务。

请注意，正在运行的容器中的特定端口可能有多个发现名称。

`TargetDiscoveryName, ServiceName, ClusterName`

此维度会将您请求的流量指标数据筛选到指定 Service Connect 发现名称，但仅计入来自此服务在此集群中创建的任务的流量。

使用此维度可以查看来自其他服务中特定客户端的入站流量指标。

与 DiscoveryName, ServiceName, ClusterName 不同的是，这些流量指标仅衡量来自在此命名空间中具有 Service Connect 配置的其他 Amazon ECS 任务的传送到该 DiscoveryName 的入站流量。这包括具有仅限客户端配置或客户端/服务器 Service Connect 配置的服务执行的任务。

请注意，正在运行的容器中的特定端口可能有多个发现名称。

AWS Fargate 使用情况指标

您可以使用 CloudWatch 用量指标来提供账户资源使用情况的可见性。这些指标可在 CloudWatch 图表和控制面板上直观呈现当前的服务使用情况。

AWS Fargate 用量指标与 AWS 服务配额对应。您可以配置警报，以在用量接近服务限额时向您发出警报。有关 Fargate 服务配额的更多信息，请参阅 AWS Fargate 服务限额。

AWS Fargate 在 AWS/Usage 命名空间中发布以下指标。

指标	描述
ResourceCount	您账户中运行的指定资源的总数量。资源由与指标关联的维度定义。

以下维度用于优化由 AWS Fargate 发布的用量指标。

维度	描述
Service	包含该资源的 AWS 服务的名称。对于 AWS Fargate 用量指标，此维度的值为 Fargate。
Type	正在报告的实体的类型。目前，AWS Fargate 用量指标的唯一有效值为 Resource。
Resource	正在运行的资源的类型。正在运行的资源的类型。目前，AWS Fargate 用量指标的唯一有效值是 vCPU，它返回有关正在运行的实例的信息。
Class	所跟踪的资源的类。所跟踪的资源的类。对于以 vCPU 作为资源维度的值的 AWS Fargate 用量指标，有效值为 Standard/OnDemand 和 Standard/Spot 。

您可以使用 Service Quotas 控制台在图表上可视化您的用量，并配置警报以便在您的 AWS Fargate 用量接近服务配额时提醒您。有关如何创建 CloudWatch 警报以便在您接近配额值阈值时通知您的信息，请参阅《Service Quotas User Guide》中的 Service Quotas and Amazon CloudWatch

.

Amazon ECS 集群预留指标

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。此指标仅用于包含在 EC2 实例上托管的任务或服务的集群。包含在 AWS Fargate 上托管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)
```

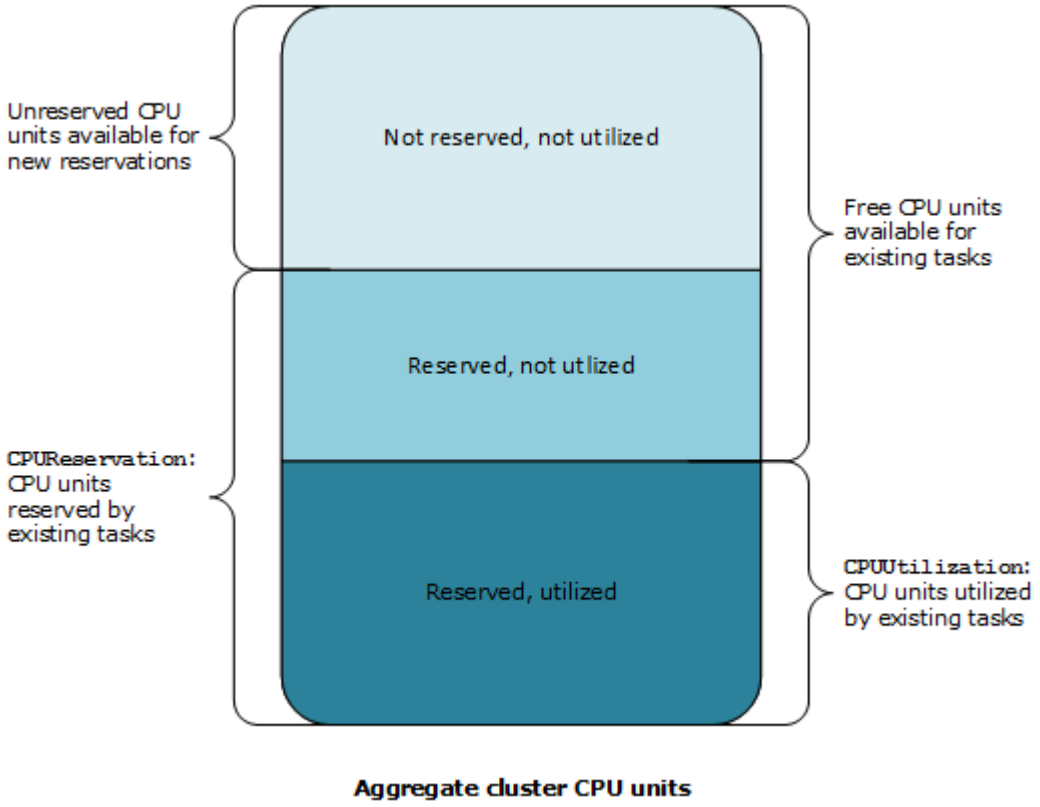
```
                                (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ---------------------------------------------------------------------
                              (Total GPUs registered by container instances in cluster)
```

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。如果您在任务定义中指定软限制（memoryReservation），它将用于计算预留内存量。否则将使用硬限制 (memory)。集群中的任务预留的总内存量（以 MiB 为单位）还包括临时文件系统（tmpfs）卷大小，如果在任务定义中定义，还包括 sharedMemorySize。有关软限制和硬限制、共享内存大小和 tmpfs 卷大小的更多信息，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 任务定义参数文档。



Amazon ECS 集群利用率指标

集群利用率指标可用于 CPU、内存以及 EBS 文件系统利用率（条件是您的任务附加有 EBS 卷）。这些指标仅适用于具有在 Amazon EC2 实例上托管的任务或服务的集群。包含在 AWS Fargate 上托管的任务的集群不支持这些指标。

Amazon ECS 集群级别的 CPU 和内存利用率指标

CPU 和内存利用率以所有任务在某个集群上使用的 CPU 和内存占为该集群中每个 Amazon EC2 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 Amazon EC2 实例将影响集群利用率指标。

```
                                  (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 Amazon EC2 实例上的 Amazon ECS 容器代理每分钟都会计算当前用于该 Amazon EC2 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 Amazon ECS 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 1,024 个 CPU 单元和 2,048 MiB 内存，则该集群上总共使用了 10,240 个 CPU 单元和 20,480 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。

Amazon ECS 集群级别 Amazon EBS 文件系统利用率

集群级别的 EBS 文件系统利用率指标是通过将集群上运行的任务所使用的 EBS 文件系统的总量，除以为集群中的所有任务分配的 EBS 文件系统存储总量而计算得到的。

```
                                     (Total GB of EBS filesystem used by tasks in
 cluster x 100)
```

```
Cluster EBS filesystem utilization =
 ------------------------------------------------------------
                                      (Total GB of EBS filesystem allocated to tasks
 in cluster)
```

Amazon ECS 服务利用率指标

服务利用率指标可用于 CPU、内存以及 EBS 文件系统利用率（条件是您的任务附加有 EBS 卷）。服务级别的指标仅支持具有 Amazon EC2 实例和 Fargate 上托管的任务的服务。

服务级别的 CPU 和内存利用率

CPU 和内存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。

```
                                    (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)
```

每分钟，Amazon ECS 容器代理都会计算当前用于服务所拥有的每个任务的 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`)。有关硬限制和软限制的更多信息，请参阅任务大小。

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%。

如果您的应用程序短时间内存利用率突然飙升，您将看不到服务内存利用率的提高，因为 Amazon ECS 每分钟收集多个数据点，然后将它们聚合发送到 CloudWatch 的一个数据点。

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。

> **Note**
>
> 在此示例中，在容器级别定义 CPU 单位时，CPU 使用率只会超过 100%。如果在任务级别定义 CPU 单元，利用率将不会超过定义的任务级别限制。

服务级别的 EBS 文件系统利用率

服务级别的 EBS 文件系统利用率是通过将属于该服务的任务使用的 EBS 文件系统的总量除以为属于该服务的所有任务分配的 EBS 文件系统存储总量计算得到的。

```
                                    (Total GB of EBS filesystem used by tasks in the
 service x 100)
Service EBS filesystem utilization =
 ------------------------------------------------------------------
                                    (Total GB of EBS filesystem allocated to tasks
 in the service)
```

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。

Amazon ECS CloudWatch Container Insights 中的服务 **RUNNING** 任务计数

使用 Amazon ECS CloudWatch Container Insights 时，每个集群和每个服务都有一个“运行任务数”（`RunningTaskCount`）指标。可以通过选择加入 `containerInsights` 账户设置来为创建的所有新集群使用 Container Insights，通过在集群创建期间启用集群设置来对单个集群使用它，或者通过

UpdateClusterSettings API 来对现有集群使用它。CloudWatch Container Insights 收集的指标按自定义指标收费。有关 CloudWatch 定价的信息，请参阅 CloudWatch 定价。

要查看此指标，请参阅《Amazon CloudWatch 用户指南》中的 Amazon ECS Container Insights 指标。

使用 EventBridge 自动响应 Amazon ECS 错误

使用 Amazon EventBridge，您可以自动执行 AWS 服务并自动响应系统事件，例如应用程序可用性问题或资源更改。AWS 服务中的事件将近乎实时传输到 EventBridge。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。可自动配置的操作包括以下操作：

- 将事件添加到 CloudWatch Logs 中的日志组
- 调用 AWS Lambda 函数
- 调用 Amazon EC2 Run Command
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知 Amazon SNS 主题或 Amazon Simple Queue Service (Amazon SQS) 队列

有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 入门。

您可以使用 EventBridge 的 Amazon ECS 事件来接收有关 Amazon ECS 集群当前状态的近实时通知。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。如果您的任务使用 Fargate 启动类型，您可以看到容器实例的状态。

使用 EventBridge，您可以基于 Amazon ECS 构建负责跨集群编排任务并近实时监控这些集群的状态的自定义计划程序。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 EventBridge 目标以异步方式处理 Amazon ECS 状态更改。目标可能包括 AWS Lambda、Amazon Simple Queue Service、Amazon Simple Notification Service 或 Amazon Kinesis Data Streams。

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 Amazon ECS 事件。

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 Events

- 处理 Amazon ECS 事件

Amazon ECS Events

Amazon ECS 跟踪每个任务和服务的状态。如果任务或服务的状态发生更改，则会生成事件并将其发送到 Amazon EventBridge。这些事件归类为任务状态更改事件和服务操作事件。以下部分中更详细地介绍了这些事件及可能原因。

Amazon ECS 生成并将以下类型的事件发送到 EventBridge：容器实例状态更改事件、任务状态更改事件、服务操作和服务部署状态更改事件。

- 容器实例状态更改事件
- 任务状态更改
- 部署状态更改
- 服务操作

> **Note**
>
> Amazon ECS 将来可能会增加其他事件类型、源和详细信息。如果您以代码方式对事件 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，将为同一活动生成多个事件。例如，在容器实例上启动任务时，将为新任务生成任务状态更改事件。生成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生成事件。

容器状态更改和任务状态更改事件包含两个 `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。下面介绍了这两个字段之间的差异：

- 对于所有事件，事件主体中的 `version` 字段设为 `0`。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。
- 事件的 `detail` 对象中的 `version` 字段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。重复事件在 `detail` 对象中具有相同版本。如果您用 EventBridge 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资源的 EventBridge 事件中报告的版本（在 `detail` 对象中），以便验证事件流中的版本是否为最新版本。

服务操作事件仅包含主体中的 `version` 字段。

有关如何集成 Amazon ECS 和 EventBridge 的更多信息，请参阅 Integrating Amazon EventBridge and Amazon ECS（集成 Amazon EventBridge 和 Amazon ECS）。

Amazon ECS 容器实例状态更改事件

以下方案将引起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 Management Console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 服务调度器启动或停止任务。

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）。针对状态为 `RUNNING` 的任务，Amazon ECS 容器代理调用状态为 `STOPPED` 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 `STOPPED` 的任务转换为 `RUNNING`，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 Management Console或开发工具包调用）注销容器实例。

注销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。EC2 实例停止后，任务也停止。

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 Container 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 容器代理每小时断开连接并重新连接多次，这是其正常操作的一部分，因此代理连接事件应该是可以预期的。这些事件并不表示容器代理或容器实例存在问题。

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生成事件。

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形式传送。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 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

```
{
  "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"
  }
}
```

Amazon EC2 任务状态更改事件

以下方案将引起任务状态更改事件：

您调用 StartTask、RunTask 或 StopTask API 操作（直接调用，或者通过 AWS Management Console、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 服务调度器启动或停止任务。

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。

对于 Amazon EC2 启动类型，Amazon ECS 容器代理将监控容器实例上的任务的状态。Amazon ECS 容器代理将报告任何状态更改。状态更改可能包括从 PENDING 到 RUNNING 或从 RUNNING 到 STOPPED 的更改。

您可以使用 DeregisterContainerInstance API 操作和 force 标志（直接调用，或者通过 AWS Management Console或开发工具包调用）强制取消注册基础容器实例。

注销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生成事件。

使用 Fargate Spot 容量提供程序的任务会收到终止通知。

当任务正在使用 FARGATE_SPOT 容量提供程序并且由于 Spot 中断而停止时会生成任务状态更改事件。

Example 任务状态更改事件

任务状态更改事件以下面的形式传送。下面的 `detail` 部分类似于 Amazon Elastic Container Service API Reference 中的 DescribeTasks API 操作返回的任务对象。如果您的容器使用通过 Amazon ECR 托管的映像，则返回 `imageDigest` 字段。

> **Note**
>
> `createdAt`、`connectivityAt`、`pullStartedAt`、`startedAt`、`pullStoppedAt` 和 `updatedAt` 字段的值是 `DescribeTasks` 操作的响应中的 UNIX 时间戳，而在任务状态更改事件中，它们是 ISO 字符串时间戳。

有关 CloudWatch Events 参数的更多信息，请参阅 Amazon EventBridge 用户指南中的事件和事件模式。

有关如何配置一个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的 Amazon EventBridge 事件规则，请参阅 针对 Amazon ECS 任务停止事件发送 Amazon Simple Notification Service 警报

```
{
    "version": "0",
    "id": "3317b2af-7005-947d-b652-f55e762e571a",
    "detail-type": "ECS Task State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-01-23T17:57:58Z",
    "region": "us-west-2",
    "resources": [
```

```
        "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad"
    ],
    "detail": {
        "attachments": [
            {
                "id": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-abcd1234"
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-abcd1234"
                    },
                    {
                        "name": "macAddress",
                        "value": "0a:98:eb:a7:29:ba"
                    },
                    {
                        "name": "privateIPv4Address",
                        "value": "10.0.0.139"
                    }
                ]
            }
        ],
        "availabilityZone": "us-west-2c",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/FargateCluster",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-west-2:111122223333:container/
cf159fd6-3e3f-4a9e-84f9-66cbe726af01",
                "lastStatus": "RUNNING",
                "name": "FargateApp",
                "image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/hello-
repository:latest",
                "imageDigest":
 "sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6",
                "runtimeId":
 "ad64cbc71c7fb31c55507ec24c9f77947132b03d48d9961115cf24f3b7307e1e",
```

```
                "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
                "networkInterfaces": [
                    {
                        "attachmentId": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                        "privateIpv4Address": "10.0.0.139"
                    }
                ],
                "cpu": "0"
            }
        ],
        "createdAt": "2020-01-23T17:57:34.402Z",
        "launchType": "FARGATE",
        "cpu": "256",
        "memory": "512",
        "desiredStatus": "RUNNING",
        "group": "family:sample-fargate",
        "lastStatus": "RUNNING",
        "overrides": {
            "containerOverrides": [
                {
                    "name": "FargateApp"
                }
            ]
        },
        "connectivity": "CONNECTED",
        "connectivityAt": "2020-01-23T17:57:38.453Z",
        "pullStartedAt": "2020-01-23T17:57:52.103Z",
        "startedAt": "2020-01-23T17:57:58.103Z",
        "pullStoppedAt": "2020-01-23T17:57:55.103Z",
        "updatedAt": "2020-01-23T17:57:58.103Z",
        "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
        "taskDefinitionArn": "arn:aws:ecs:us-west-2:111122223333:task-definition/
sample-fargate:1",
        "version": 4,
        "platformVersion": "1.3.0"
    }
}
```

Amazon ECS 服务操作事件

Amazon ECS 发送具有详细信息类型 ECS Service Action 的服务操作事件。与容器实例和任务状态更改事件不同，服务操作事件在 `details` 响应字段中不包含版本号。以下是用于为 Amazon ECS 服务操作事件创建 EventBridge 规则的事件模式。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建 EventBridge 规则

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Service Action"
    ]
}
```

Amazon ECS 发送具有 `INFO`、`WARN`, and `ERROR` 事件类型的事件。以下是服务操作事件。

具有 **`INFO`** 事件类型的服务操作事件

`SERVICE_STEADY_STATE`

服务正常运行且任务数量为期望值，从而达到稳定状态。服务调度器会定期报告状态，因此您可能会多次收到此消息。

`TASKSET_STEADY_STATE`

任务集正常且任务数量为期望值，从而达到稳定状态。

`CAPACITY_PROVIDER_STEADY_STATE`

与服务关联的容量提供程序达到稳定状态。

`SERVICE_DESIRED_COUNT_UPDATED`

当服务计划程序为服务或任务集更新计算的期望计数时。当用户手动更新期望计数时，不会发送此事件。

具有 **`WARN`** 事件类型的服务操作事件

`SERVICE_TASK_START_IMPAIRED`

该服务无法始终如一地成功启动任务。

`SERVICE_DISCOVERY_INSTANCE_UNHEALTHY`

使用服务发现的服务包含运行状况不佳的任务。服务计划程序检测到服务注册表中的任务运行状况不佳。

具有 **ERROR** 事件类型的服务操作事件

`SERVICE_DAEMON_PLACEMENT_CONSTRAINT_VIOLATED`

使用 DAEMON 服务计划程序策略的服务中的任务不再符合服务的放置约束策略。

`ECS_OPERATION_THROTTLED`

由于 Amazon ECS API 节流限制，服务计划程序已被限制。

`SERVICE_DISCOVERY_OPERATION_THROTTLED`

由于 AWS Cloud Map API 节流限制，服务计划程序已被限制。配置为使用服务发现的服务可能会发生这种情况。

`SERVICE_TASK_PLACEMENT_FAILURE`

服务计划程序无法放置任务。原因将在 `reason` 字段中描述。

生成此服务事件的常见原因是集群中缺乏放置任务的资源。例如，可用容器实例上的 CPU 或内存容量不足，或者没有容器实例可用。另一个常见原因是 Amazon ECS 容器代理在容器实例上断开连接，导致计划程序无法放置任务。

`SERVICE_TASK_CONFIGURATION_FAILURE`

由于配置错误，服务计划程序无法放置任务。原因将在 `reason` 字段中描述。

生成此服务事件的一个常见原因是，标签已应用于服务，但用户或角色尚未在区域中选择采用新的 Amazon 资源名称（ARN）格式。有关更多信息，请参阅 Amazon Resource Names (ARN) 和 ID。另一个常见原因是 Amazon ECS 无法承担 IAM 角色所提供的任务。

Example 服务稳定状态事件

服务稳定状态事件以下面的形式传送。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

```
{
    "version": "0",
```

```
    "id": "af3c496d-f4a8-65d1-70f4-a69d52e9b584",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:27:22Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:27:22.695Z"
    }
}
```

Example 容量提供程序稳定状态事件

容量提供程序稳定状态事件以下面的形式传送。

```
{
    "version": "0",
    "id": "b9baa007-2f33-0eb1-5760-0d02a572d81f",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:37:00Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "CAPACITY_PROVIDER_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/ASG-tutorial-
capacity-provider"
        ],
        "createdAt": "2019-11-19T19:37:00.807Z"
    }
}
```

Example 服务任务启动受损事件

服务任务启动受损事件以下面的形式传送。

```
{
    "version": "0",
    "id": "57c9506e-9d21-294c-d2fe-e8738da7e67d",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "WARN",
        "eventName": "SERVICE_TASK_START_IMPAIRED",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:55:38.725Z"
    }
}
```

Example 服务任务放置失败事件

服务任务放置失败事件以下面的形式传送。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

在以下示例中，任务试图使用 FARGATE_SPOT 容量提供程序，但服务计划程序无法获取任何 Fargate Spot 容量。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
```

```
    "detail": {
        "eventType": "ERROR",
        "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/FARGATE_SPOT"
        ],
        "reason": "RESOURCE:FARGATE",
        "createdAt": "2019-11-06T19:09:33.087Z"
    }
}
```

在以下 EC2 启动类型的示例中，任务尝试在容器实例 2dd1b186f39845a584488d2ef155c131 上启动，但是由于 CPU 不足，服务调度器无法放置任务。

```
{
  "version": "0",
  "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
  "detail-type": "ECS Service Action",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2019-11-19T19:55:38Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
  ],
  "detail": {
    "eventType": "ERROR",
    "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
    "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
    ],
    "reason": "RESOURCE:CPU",
    "createdAt": "2019-11-06T19:09:33.087Z"
  }
}
```

Amazon ECS 服务部署状态更改事件

Amazon ECS 发送详细信息类型为 ECS部署状态更改的服务部署更改状态事件。以下是用于为 Amazon ECS 任务状态更改事件创建 EventBridge 规则的事件模式。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建 EventBridge 规则

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Deployment State Change"
    ]
}
```

Amazon ECS 发送具有INFO 和 ERROR 事件类型的事件。下列是服务部署状态更改的事件：

`SERVICE_DEPLOYMENT_IN_PROGRESS`

正在进行服务部署。此事件同时针对初始部署和回滚部署发送。

`SERVICE_DEPLOYMENT_COMPLETED`

服务部署已完成。一旦服务在部署后达到稳定状态，就会发送此事件。

`SERVICE_DEPLOYMENT_FAILED`

服务部署失败。将针对启用了部署断路器逻辑的服务发送此事件。

Example 正在进行的服务部署事件

初始部署和回滚部署启动时，将传递服务部署进行中的事件。两者之间的差异在 `reason` 字段中。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

以下显示了初始部署开始的示例输出。

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3a6EXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
```

```
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment deploymentId in progress."
    }
}
```

下面显示了开始回滚部署的示例输出。reason 字段提供服务正在回滚到部署的 ID。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: rolling back to
 deploymentId deploymentID."
    }
}
```

Example 服务部署已完成事件

服务部署完成状态事件以下面的形式传送。有关更多信息，请参阅 通过替换任务来部署 Amazon ECS 服务。

```
{
```

```
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_COMPLETED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment deploymentID completed."
    }
}
```

Example 服务部署失败事件

服务部署失败状态事件以下面的形式传送。将仅针对启用了部署断路器逻辑的服务发送服务部署失败状态事件。有关更多信息，请参阅 通过替换任务来部署 Amazon ECS 服务。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "ERROR",
         "eventName": "SERVICE_DEPLOYMENT_FAILED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: task failed to start."
    }
}
```

处理 Amazon ECS 事件

Amazon ECS 至少发送一次事件。这意味着您可能会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。

为了正确排序事件，每个事件的 `detail` 部分均包含 `version` 属性。每次当资源更改状态时，此 `version` 会递增。重复事件在 `detail` 对象中具有相同 `version`。如果使用 EventBridge 复制 Amazon ECS 容器实例和任务状态，则可以将 Amazon ECS API 报告的 `version` 资源版本与资源在 EventBridge 中报告的版本进行比较，以验证事件流中的版本是否为当前版本。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。

示例：在 AWS Lambda 函数中处理事件

以下示例显示了一个用 Python 3.9 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
        )
```

以下 Fargate 示例显示了一个用 Python 3.9 编写的 Lambda 函数，此函数可捕获任务状态更改事件，并将这些事件保存到以下 Amazon DynamoDB 表：

```
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"]
    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
        )
```

使用 Container Insights 监控 Amazon ECS 容器

CloudWatch Container Insights 从容器化应用程序和微服务中收集、聚合及汇总指标与日志。

Container Insights 将发现集群中所有正在运行的容器，并在每个性能堆栈层中收集性能数据。运行数据是作为性能日志事件收集的。这些条目使用结构化 JSON 模式来大规模摄取和存储高基数数据。从该数据中，CloudWatch 在集群、服务和任务级别创建更高级别的聚合指标以作为 CloudWatch 指标。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。有关可用指标的信息，请参阅《Amazon CloudWatch 用户指南》中的 Amazon ECS Container Insights 指标。

> **Important**
>
> CloudWatch Container Insights 收集的指标按自定义指标收费。有关 CloudWatch 定价的信息，请参阅 CloudWatch 定价。Amazon ECS 还提供了不产生额外成本的监控指标。有关更多信息，请参阅 使用 CloudWatch 监控 Amazon ECS 。

注意事项

在使用 CloudWatch Container Insights时，应考虑以下事项。

- CloudWatch Container Insights 指标仅反映指定时间范围内具有正在运行的任务的资源。例如，如果您的集群包含一个服务，但该服务没有处于 `RUNNING` 状态的任务，则不会向 CloudWatch 发送任何指标。如果您有两个服务，其中一个服务具有正在运行的任务，而另一个服务没有，则仅发送具有正在运行的任务的服务的指标。
- 网络指标适用于在 Fargate 上运行的所有任务以及在 Amazon EC2 实例上运行的任务，这些任务使用 `bridge` 或 `awsvpc` 网络模式。

您可以在 CloudWatch Container Insights 控制台中查看 Amazon ECS 任务和服务生命周期事件。这样有助于您在一个视图中将容器指标、日志和事件关联在一起，从而便于您更完整地了解运行情况。

您可以查看的事件为 Amazon ECS 发送给 Amazon EventBridge 的事件。有关更多信息，请参阅 Amazon ECS 事件。

您可以选择为集群、任务或服务配置性能指标。根据您选择的资源，系统会报告以下事件：

- 容器实例状态更改事件
- 服务操作事件
- 任务状态更改事件

为 Amazon ECS 配置 CloudWatch Container Insights

您可以使用 Amazon ECS 控制台、AWS CLI、API 和 SDK 配置 Container Insights。

使用下表确定添加 Container Insights 所需采取的操作。

Amazon ECS 资源标记支持

任务	控制台	AWS CLI	API 操作
更改所有用户的默认设置	修改 Amazon ECS 账户设置	put-account-setting-default	PutAccountSettingDefault
更改特定用户的默认设置	修改 Amazon ECS 账户设置	put-account-setting	PutAccountSetting
为特定集群配置 Container Insights	创建 Fargate 启动类型的 Amazon ECS 集群 为 Amazon EC2 启动类型创建 Amazon ECS 集群 更新 Amazon ECS 集群	create-cluster UpdateCluster	CreateCluster UpdateCluster

Important

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 Linux 容器实例管理。

CloudWatch Container Insights 查看 Amazon ECS 生命周期事件所需的权限

您必须先配置正确的权限，然后才能在 CloudWatch Container Insights 控制台中配置和查看事件。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的 Container Insights 中的 Amazon ECS 生命周期事件。有关 CloudWatch 的 IAM policy 的更多信息，请参阅 CloudWatch 的 AWS Identity and Access Management。

配置 Container Insights 以查看 Amazon ECS 生命周期事件所需的权限

在任务角色中，配置生命周期事件需要以下权限：

- events:PutRule

- events:PutTargets
- logs:CreateLogGroup

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "logs:CreateLogGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

在 Container Insights 中查看 Amazon ECS 生命周期事件所需的权限

查看生命周期事件需要以下权限。将以下权限作为内联策略添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM policy。

- events:DescribeRule
- events:ListTargetsByRule
- logs:DescribeLogGroups

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

使用容器运行状况检查确定 Amazon ECS 任务运行状况

当创建任务定义时，您可以为容器配置运行状况检查。运行状况检查指的是在容器上本地运行的命令，可验证应用程序的运行状况和可用性。

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。

在任务定义中定义运行状况检查时，容器会在容器内部运行运行状况检查进程，然后评估退出代码以确定应用程序的运行状况。

运行状况检查包含以下参数：

- 命令 – 容器运行的命令，用于确定其运行状况是否正常。字符串数组可以以 CMD 开头以直接运行命令参数，或者以 CMD-SHELL 开头以使用容器的默认 Shell 来运行命令。
- 间隔 – 每次运行状况检查间隔的时间（以秒为单位）。
- 超时 – 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。
- 重试次数 – 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。
- 启动期 – 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。

有关如何在任务定义中指定运行状况检查的信息，请参阅运行状况检查。

下面介绍了容器的运行状况可能值：

- HEALTHY – 容器运行状况检查已成功通过。
- UNHEALTHY – 容器运行状况检查失败。
- UNKNOWN – 正在评估容器运行状况检查，没有定义容器运行状况检查，或者 Amazon ECS 没有容器的运行状况。

运行状况检查命令在容器上运行。因此，您必须在容器映像中包含这些命令。

运行状况检查通过位于 `localhost` 或 `127.0.0.1` 的容器的环回接口连接到应用程序。退出代码 `0` 表示成功，非零退出代码表示失败。

使用容器运行状况检查时，请考虑以下几点：

- 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。
- 如果使用的是 Linux 平台版本 `1.1.0` 或更高版本或者 Windows 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

Amazon ECS 如何确定任务运行状况

只有在任务定义中具有运行状况检查命令的必要容器才会被考虑用于确定任务运行状况。

以下规则按顺序评估：

1. 如果一个基本容器的状态为 UNHEALTHY，则任务状态为 UNHEALTHY。
2. 如果一个基本容器的状态为 UNKNOWN，则任务状态为 UNKNOWN。
3. 如果所有基本容器的状态都为 HEALTHY，则任务状态为 HEALTHY。

请考虑以下任务运行状况示例，其中包含 2 个基本容器。

容器 1 运行状况	容器 2 运行状况	任务运行状况
UNHEALTHY	UNKNOWN	UNHEALTHY
UNHEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY

考虑以下任务运行状况示例，其中包含 3 个容器。

容器 1 运行状况	容器 2 运行状况	容器 3 运行状况	任务运行状况
UNHEALTHY	UNKNOWN	UNKNOWN	UNHEALTHY
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY	HEALTHY	UNHEALTHY

容器 1 运行状况	容器 2 运行状况	容器 3 运行状况	任务运行状况
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

代理断开连接如何影响运行状况检查

如果 Amazon ECS 容器代理与 Amazon ECS 服务断开连接，这不会导致容器转变为 UNHEALTHY 状态。这是设计使然，目的是确保容器在代理重启或暂时不可用时保持运行。运行状况检查状态是 Amazon ECS 代理的“最后一次收到位置”的响应，因此，如果在断开连接之前将容器视为 HEALTHY，则该状态将保持不变，直到代理重新连接并进行另一次运行状况检查。对容器运行状况检查的状态未做出任何假设。

查看 Amazon ECS 容器运行状况

您可以在控制台中和使用 DescribeTasks 响应中的 API 查看容器运行状况。有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 DescribeTasks。

如果您对容器（例如 Amazon CloudWatch Logs）使用日志记录，则可以将运行状况检查命令配置为将容器运行状况输出转发到您的日志。请确保使用 2&1 同时捕获 stdout 和 stderr 信息。

```
"command": [
     "CMD-SHELL",
     "curl -f http://localhost/ >> /proc/1/fd/1 2>&1  || exit 1"
   ],
```

监控 Amazon ECS 容器实例运行状况

Amazon ECS 提供容器实例运行状况监控。您可以快速确定 Amazon ECS 是否已经检测到可能阻止您的容器实例运行容器的任何问题。Amazon ECS 对每个使用代理版本 1.57.0 或更高版本运行的容器实例执行自动检查，以确定问题。有关验证代理版本和容器实例的更多信息，请参阅 更新 Amazon ECS 容器代理。

您必须使用 AWS CLI 版本 1.22.3 或更高版本，或者 AWS CLI 版本 2.3.6 或更高版本。有关如何更新 AWS CLI 的信息，请参阅 AWS Command Line Interface 用户指南版本 2 中的安装或更新 AWS CLI 的最新版本。

状态检查每分钟进行两次，会返回一个通过或失败状态。如果所有的检查都通过，则实例的整体状态是 OK。如果有一个或多个检查故障，则整体状态为 IMPAIRED。状态检查内置在 Amazon ECS 容器代理中，所以不能关闭或删除。您可以通过查看这些状态检查的结果来识别特定的和可检测的问题。有关更多信息，请参阅 the section called “运行状况检查”。

使用 CONTAINER_INSTANCE_HEALTH 选项运行 DescribeContainerInstances API，以检索容器实例运行状况。

```
aws ecs describe-container-instances \
     --cluster cluster_name \
     --container-instances 47279cd2cadb41cbaef2dcEXAMPLE \
     --include CONTAINER_INSTANCE_HEALTH
```

下面是输出中运行状况对象的示例。

```
"healthStatus": {
 "overallStatus": "OK",
 "details": [{
  "type": "CONTAINER_RUNTIME",
  "status": "OK",
  "lastUpdated": "2021-11-10T03:30:26+00:00",
  "lastStatusChange": "2021-11-10T03:26:41+00:00"
 }]
}
```

相关 主题

- 使用 CloudWatch 监控 Amazon ECS

使用应用程序跟踪数据来识别 Amazon ECS 的优化机会

Amazon ECS 与 AWS Distro for OpenTelemetry 集成，以从应用程序中收集跟踪数据。Amazon ECS 使用 AWS Distro for OpenTelemetry 附加容器，以收集跟踪数据并将其路由到 AWS X-Ray。有关更多信息，请参阅在 Amazon ECS 中设置 AWS Distro for OpenTelemetry Collector。然后，您可以使用 AWS X-Ray 来识别错误和异常、分析性能瓶颈和响应时间。

要使 AWS Distro for OpenTelemetry Collector 将跟踪数据发送到 AWS X-Ray，必须将您的应用程序配置为创建跟踪数据。有关更多信息，请参阅 AWS X-Ray 开发人员指南中的分析 AWS X-Ray 的应用程序。

AWS Distro for OpenTelemetry 与 AWS X-Ray 集成所需的 IAM 权限

Amazon ECS 与 AWS Distro for OpenTelemetry 集成要求您创建任务角色并在任务定义中指定角色。我们建议您为配置 AWS Distro for OpenTelemetry sidecar，以将容器日志路由到 CloudWatch Logs。

> **Important**
>
> 如果您还要使用 AWS Distro for OpenTelemetry 集成收集应用程序指标，请确保您的任务 IAM 角色还包含该集成所需的权限。有关更多信息，请参阅 使用应用程序指标关联 Amazon ECS 应用程序性能。

创建以下策略，之后将其附加到任务执行角色。

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
```

```
                "logs:PutRetentionPolicy",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

指定 AWS Distro for OpenTelemetry 附加，用于任务定义中的 AWS X-Ray 集成

Amazon ECS 控制台通过使用使用跟踪收集选项简化创建 AWS Distro for OpenTelemetry 附加容器的过程。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

如果您没有使用 Amazon ECS 控制台，则可以添加 AWS Distro for OpenTelemetry 附加容器到您的任务定义中。以下任务定义片段显示了用于添加 AWS Distro for OpenTelemetry 附加以进行 AWS X-Ray 集成的容器定义。

```
{
 "family": "otel-using-xray",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryXrayRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
```

```
    "name": "aws-otel-emitter",
    "image": "application-image",
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/aws-otel-emitter",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    },
    "dependsOn": [{
     "containerName": "aws-otel-collector",
     "condition": "START"
    }]
   },
   {
    "name": "aws-otel-collector",
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
    "essential": true,
    "command": [
     "--config=/etc/ecs/otel-instance-metrics-config.yaml"
    ],
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "us-east-1",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
}
```

使用应用程序指标关联 Amazon ECS 应用程序性能

Fargate 上的 Amazon ECS 支持从 Fargate 上运行的应用程序中收集指标，并将其导出到 Amazon CloudWatch 或 Amazon Managed Service for Prometheus。

您可以使用收集的元数据将应用程序性能数据与底层基础设施数据关联起来，从而缩短解决问题的平均时间。

Amazon ECS 使用 AWS Distro for OpenTelemetry 附加容器，以收集应用程序指标并将其路由到目的地。Amazon ECS 控制台体验简化了在创建任务定义时添加此集成的过程。

主题

- 将应用程序指标导出到 Amazon CloudWatch
- 将应用程序指标导出到 Amazon Managed Service for Prometheus

将应用程序指标导出到 Amazon CloudWatch

Fargate 上的 Amazon ECS 支持将您的自定义应用程序指标作为自定义指标导出到 Amazon CloudWatch。此操作通过添加 AWS Distro for OpenTelemetry 附加容器到您的任务定义中完成。Amazon ECS 控制台通过在创建新的任务定义时添加使用指标收集选项来简化此流程。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

应用程序指标将使用日志组名称 `/aws/ecs/application/metrics` 导出到 CloudWatch Logs 并且指标可在 `ECS/AWSOTel/Application` 命名空间中查看。您的应用程序必须使用 OpenTelemetry 开发工具包进行分析。有关更多信息，请参阅 AWS Distro for OpenTelemetry 文档中的 AWS Distro for OpenTelemetry 的介绍。

注意事项

使用 Fargate 上的 Amazon ECS 与 AWS Distro for OpenTelemetry 集成将应用程序指标发送到 Amazon CloudWatch 时，应考虑以下因素。

- 此集成仅将您的自定义应用程序指标发送到 CloudWatch。如果您需要任务级指标，可以在 Amazon ECS 集群配置中启用 Container Insights。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。
- Fargate 和 Amazon EC2 实例上托管的 Amazon ECS 工作负载支持 AWS Distro for OpenTelemetry 集成。目前不支持外部实例。

- CloudWatch 对每个指标最多支持 30 个维度。预设情况下，Amazon ECS 默认将 `TaskARN`、`ClusterARN`、`LaunchType`、`TaskDefinitionFamily` 和 `TaskDefinitionRevision` 维度包含到指标中。其余 25 个维度可以由您的应用程序定义。如果配置了超过 30 个维度，CloudWatch 将无法显示它们。发生这种情况时，应用程序指标将出现在 `ECS/AWSOTel/Application` CloudWatch 指标命名空间中，但没有任何维度。您可以对应用程序进行分析以添加其他维度。有关更多信息，请参阅 AWS Distro for OpenTelemetry 文档中的将 CloudWatch 指标与 AWS Distro for OpenTelemetry 结合使用。

AWS Distro for OpenTelemetry 与 Amazon CloudWatch 集成所需的 IAM 权限

Amazon ECS 与 AWS Distro for OpenTelemetry 集成要求您创建任务 IAM 角色并在任务定义中指定角色。我们建议也将 AWS Distro for OpenTelemetry 附加配置为将容器日志路由到 CloudWatch Logs 中，这还需要在任务定义中创建和指定任务执行 IAM 角色。Amazon ECS 控制台代表您处理任务执行 IAM 角色，但必须手动创建任务 IAM 角色并将其添加到您的任务定义中。有关任务执行 IAM 角色的更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

> **Important**
>
> 如果您还要使用 AWS Distro for OpenTelemetry 集成收集应用程序跟踪数据，请确保您的任务 IAM 角色还包含该集成所需的权限。有关更多信息，请参阅 使用应用程序跟踪数据来识别 Amazon ECS 的优化机会。
>
> 如果您的应用程序需要任何其他权限，则应将其添加到此策略中。每个任务定义只能指定一个任务 IAM 角色。例如，如果您使用存储在 Systems Manager 中的自定义配置文件，则应添加 `ssm:GetParameters` 权限到此 IAM policy 中。

要创建 Elastic Container Service 的服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，选择 Elastic Container Service，然后选择 Elastic Container Service 任务使用案例。
5. 选择下一步。
6. 在添加权限部分中，搜索 AWSDistroOpenTelemetryPolicyForXray，然后选择策略。

7. （可选）设置权限边界。这是一项高级特征，可用于服务角色，但不可用于服务相关角色。

   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包括您的账户中的 AWS 托管式策略和客户管理型策略的列表。

   b. 选择要用于权限边界的策略。

8. 选择下一步。
9. 输入有助于识别角色的作用的角色名称或者角色名称后缀。

   > **Important**
   >
   > 命名角色时，请注意以下事项：
   >
   > - 角色名称在您的 AWS 账户 中必须是唯一的，且不能因大小写而变得唯一。
   >
   >   例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。
   > - 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。

10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。
12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。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 资源。
13. 检查该角色，然后选择创建角色。

在您的任务定义中指定 AWS Distro for OpenTelemetry 附加

Amazon ECS 控制台通过使用使用指标收集选项简化创建 AWS Distro for OpenTelemetry 附加容器的体验。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

如果您没有使用 Amazon ECS 控制台，则可以手动添加 AWS Distro for OpenTelemetry 附加容器到您的任务定义中。以下任务定义示例显示了用于添加 AWS Distro for OpenTelemetry 附加以进行 Amazon CloudWatch 集成的容器定义。

```
{
 "family": "otel-using-cloudwatch",
```

```
"taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
"executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
"containerDefinitions": [
   {
  "name": "aws-otel-emitter",
  "image": "application-image",
  "logConfiguration": {
   "logDriver": "awslogs",
   "options": {
    "awslogs-create-group": "true",
    "awslogs-group": "/ecs/aws-otel-emitter",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "ecs"
   }
  },
  "dependsOn": [{
   "containerName": "aws-otel-collector",
   "condition": "START"
  }]
 },
 {
  "name": "aws-otel-collector",
  "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
  "essential": true,
  "command": [
   "--config=/etc/ecs/ecs-cloudwatch.yaml"
  ],
  "logConfiguration": {
   "logDriver": "awslogs",
   "options": {
    "awslogs-create-group": "True",
    "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "ecs"
   }
  }
 }
],
"networkMode": "awsvpc",
"requiresCompatibilities": [
 "FARGATE"
],
"cpu": "1024",
"memory": "3072"
```

```
}
```

将应用程序指标导出到 Amazon Managed Service for Prometheus

Amazon ECS 支持将您的任务级 CPU、内存、网络和存储指标以及自定义应用程序指标导出到 Amazon Managed Service for Prometheus。此操作通过添加 AWS Distro for OpenTelemetry 附加容器到您的任务定义中完成。Amazon ECS 控制台通过在创建新的任务定义时添加使用指标收集选项来简化此流程。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

这些指标将导出到 Amazon Managed Service for Prometheus，并可以使用 Amazon Managed Grafana 控制面板查看。您的应用程序必须使用 Prometheus 库或 OpenTelemetry 开发工具包进行分析。有关使用 OpenTelemetry 开发工具包分析您的应用程序的更多信息，请参阅 AWS Distro for OpenTelemetry 文档中的 AWS Distro for OpenTelemetry 的介绍。

使用 Prometheus 库时，您的应用程序必须公开用于抓取指标数据的 `/metrics` 终端节点。有关使用 Prometheus 库分析您的应用程序的更多信息，请参阅 Prometheus 文档中的 Prometheus 客户端库。

注意事项

使用 Fargate 上的 Amazon ECS 与 AWS Distro for OpenTelemetry 的集成向 Amazon Managed Service for Prometheus 发送应用程序指标时，应考虑以下因素。

- Fargate 和 Amazon EC2 实例上托管的 Amazon ECS 工作负载支持 AWS Distro for OpenTelemetry 集成。目前不支持外部实例。
- 预设情况下，在向 Amazon Managed Service for Prometheus 导出时，AWS Distro for OpenTelemetry 包括可用于您的应用程序指标的所有可用的任务级别维度。您还可以对应用程序进行分析以添加其他维度。有关更多信息，请参阅 AWS Distro for OpenTelemetry 文档中的开始将 Prometheus Remote Write Exporter 用于 Amazon Managed Service for Prometheus。

AWS Distro for OpenTelemetry 与 Amazon Managed Service for Prometheus 集成所需的 IAM 权限

将 Amazon ECS 与使用 AWS Distro for OpenTelemetry 附加的 Amazon Managed Service for Prometheus 集成要求您创建任务 IAM 角色并在任务定义中指定角色。在注册任务定义之前，必须使用以下步骤手动创建此任务 IAM 角色。

我们建议也将 AWS Distro for OpenTelemetry 附加配置为将容器日志路由到 CloudWatch Logs 中，这还需要在任务定义中创建和指定任务执行 IAM 角色。Amazon ECS 控制台代表您处理任务执行 IAM 角

色，但必须手动创建任务 IAM 角色。有关创建任务执行 IAM 角色的更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

> **Important**
>
> 如果您还要使用 AWS Distro for OpenTelemetry 集成收集应用程序跟踪数据，请确保您的任务 IAM 角色还包含该集成所需的权限。有关更多信息，请参阅 使用应用程序跟踪数据来识别 Amazon ECS 的优化机会。

要创建 Elastic Container Service 的服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，选择 Elastic Container Service，然后选择 Elastic Container Service 任务使用案例。
5. 选择下一步。
6. 在添加权限部分中，搜索 AmazonPrometheusRemoteWriteAccess，然后选择策略。
7. （可选）设置权限边界。这是一项高级特征，可用于服务角色，但不可用于服务相关角色。
   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包括您的账户中的 AWS 托管式策略和客户管理型策略的列表。

   b. 选择要用于权限边界的策略。
8. 选择下一步。
9. 输入有助于识别角色的作用的角色名称或者角色名称后缀。

> **Important**
>
> 命名角色时，请注意以下事项：
>
> - 角色名称在您的 AWS 账户 中必须是唯一的，且不能因大小写而变得唯一。

例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。

- 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。

10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。
12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。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 资源。
13. 检查该角色，然后选择创建角色。

在您的任务定义中指定 AWS Distro for OpenTelemetry 附加

Amazon ECS 控制台通过使用使用指标收集选项简化创建 AWS Distro for OpenTelemetry 附加容器的体验。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。

如果您没有使用 Amazon ECS 控制台，则可以手动添加 AWS Distro for OpenTelemetry 附加容器到您的任务定义中。以下任务定义示例显示了用于添加 AWS Distro for OpenTelemetry 附加以进行 Amazon Managed Service for Prometheus 集成的容器定义。

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
```

```
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-amp.yaml"
   ],
   "environment": [{
    "name": "AWS_PROMETHEUS_ENDPOINT",
    "value": "https://aps-workspaces.aws-region.amazonaws.com/workspaces/
ws-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
   }],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

使用 AWS CloudTrail 记录 Amazon ECS API 调用

Amazon ECS 与 AWS CloudTrail 集成，后者是一项服务，提供 Amazon ECS 中由用户、角色或 AWS 服务所采取操作的记录。CloudTrail 将对 Amazon ECS 的所有 API 调用作为事件捕获，包括来自 Amazon ECS 控制台的调用、来自代码对 Amazon ECS API 操作的调用。为了保护您的 VPC，被 VPC 端点策略拒绝但却以其他方式允许的请求不记录在 CloudTrail 中。

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 控制台中的 事件历史记录 中查看最新事件。使用由 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 用户指南》。

CloudTrail 中的 Amazon ECS 信息

在您创建账户时，您的 AWS 账户中已启用 CloudTrail。当 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。

如要持续记录 AWS 账户中的事件（包括 Amazon ECS 事件），请创建跟踪记录，CloudTrail 将使用跟踪记录向 Amazon S3 存储桶传送日志文件。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，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行动。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
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Amazon ECS 操作都由 CloudTrail 记录，并记录在 Amazon Elastic Container Service API 参考中。例如，对 `CreateService`、`RunTask` 和 `DeleteCluster` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其它 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

跟踪记录是一种配置，允许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记账条目。一个事件表示来自任何源的一个请求，包括有关请求的操作、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，因此它们不会按任何特定顺序显示。

> **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"
}
```

使用 Amazon ECS 元数据监控工作负载

您可以使用任务和容器元数据对您的工作负载进行故障排除，并根据运行时环境进行配置更改。

元数据包括以下几个类别：

- 提供有关任务运行位置信息的任务级属性。
- 容器级属性，提供 Docker ID、名称和映像详细信息。

  这提供对容器的可见性。
- 网络设置，例如 IP 地址、子网和网络模式。

  这有助于进行网络配置和故障排除。
- 任务状态和运行状况

  这可以让您知道任务是否正在运行。

您可以通过以下任何方式查看元数据：

- 容器元数据文件

  从 Amazon ECS 容器代理版本 1.15.0 开始，容器或主机容器实例中提供了各种容器元数据。通过启用此功能，您可以从容器或主机容器实例中查询有关任务、容器以及容器实例的信息。元数据文件是在主机实例上创建的，并作为Docker卷装入容器中，因此在 AWS Fargate上承载任务时不可用。

- 任务元数据终端节点

  Amazon ECS 容器代理会向每个容器注入一个环境变量，称为任务元数据端点，它提供了各种任务元数据和 Docker 统计信息添加到容器中。

- 容器自检

  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。

Amazon ECS 容器元数据文件

从 Amazon ECS 容器代理版本 1.15.0 开始，容器或主机容器实例中提供了各种容器元数据。通过启用此功能，您可以从容器或主机容器实例中查询有关任务、容器以及容器实例的信息。元数据文件是在主机实例上创建的，并作为Docker卷装入容器中，因此在 AWS Fargate上承载任务时不可用。

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 Amazon ECS 任务和映像自动清理。

主题

- 容器元数据文件位置
- 开启 Amazon ECS 容器元数据
- Amazon ECS 容器元数据文件格式

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/`*`cluster_name`*`/`*`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)。如果您并非使用经 Amazon ECS 优化的 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。

  - 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
```

开启 Amazon ECS 容器元数据

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。

如果在代理启动时将 ECS_ENABLE_CONTAINER_METADATA 设置为 true，则会为从那时起创建的任何容器创建元数据文件。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。以下是一个示例用户数据脚本，该脚本将设置此变量以及将容器实例注册到您的集群中。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_ENABLE_CONTAINER_METADATA=true
EOF
```

Amazon ECS 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 Resource Name (ARN)。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`TaskDefinitionFamily`

容器正在使用的任务定义系列的名称。

`TaskDefinitionRevision`

容器正在使用的任务定义修订版本。

`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：
>
> ```
> tail -1 /etc/hosts | 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。

`AvailabilityZone`

主机容器实例所在的可用区。

`HostPrivateIPv4Address`

容器所属任务的私有 IP 地址。

`HostPublicIPv4Address`

容器所属任务的公有 IP 地址。

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/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/2b88376d-
aba3-4950-9ddf-bcb0f388a40c",
    "TaskDefinitionFamily": "console-sample-app-static",
    "TaskDefinitionRevision": "1",
    "ContainerID": "aec2557997f4eed9b280c2efd7afccdcedfda4ac399f7480cae870cfc7e163fd",
    "ContainerName": "simple-app",
    "CreatedAt": "2023-10-08T20:09:11.44527186Z",
    "StartedAt": "2023-10-08T20:09:11.44527186Z",
    "DockerContainerName": "/ecs-console-sample-app-static-1-simple-app-
e4e8e495e8baa5de1a00",
    "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de",
    "ImageName": "httpd:2.4",
    "PortMappings": [
        {
            "ContainerPort": 80,
            "HostPort": 80,
            "BindIp": "0.0.0.0",
```

```
            "Protocol": "tcp"
        }
    ],
    "Networks": [
        {
            "NetworkMode": "bridge",
            "IPv4Addresses": ["192.0.2.0"]
        }
    ],
    "MetadataFileStatus": "READY",
    "AvailabilityZone": "us-east-1b",
    "HostPrivateIPv4Address": "192.0.2.0",
    "HostPublicIPv4Address": "203.0.113.0"
}
```

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/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
}
```

任务元数据可用于 EC2 上的 Amazon ECS 任务

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。这称为任务元数据端点。提供了以下版本：

- 任务元数据端点版本 4 — 为容器提供各种元数据和 Docker 统计信息。还可以提供网络速率数据。可用于在运行至少 `1.39.0` 版本的 Amazon ECS 容器代理的Amazon EC2 Linux 实例上启动的 Amazon ECS 任务。对于使用 `awsvpc` 网络模式的 Amazon EC2 Windows 实例，Amazon ECS 容器代理的版本必须至少为 `1.54.0`。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4。
- 任务元数据端点版本 3 — 为容器提供各种元数据和 Docker 统计信息。可用于在运行至少 `1.21.0` 版本的 Amazon ECS 容器代理的Amazon EC2 Linux 实例上启动的 Amazon ECS 任务。对于使

用 awsvpc 网络模式的 Amazon EC2 Windows 实例，Amazon ECS 容器代理的版本必须至少为 1.54.0。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 3。

- 任务元数据端点版本 2–可用于在运行至少版本 1.17.0 的 Amazon ECS 容器代理的 Amazon EC2 Linux实例上启动的 Amazon ECS 任务。对于使用 awsvpc 网络模式的 Amazon EC2 Windows 实例，Amazon ECS 容器代理的版本必须至少为 1.54.0。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 2。

如果您的 Amazon ECS 任务托管在 Amazon EC2 上，您也可以使用实例元数据服务（IMDS）端点访问任务主机元数据。以下命令在托管任务的实例中运行时，会列出主机实例的 ID。

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

您可以从端点获取的信息分为几个类别，例如 *instance-id*。有关您可以使用端点获取的不同类别的主机实例元数据的更多信息，请参阅实例元数据类别。

Amazon ECS 任务元数据端点版本 4

Amazon ECS 容器代理会向每个容器注入一个环境变量，称为任务元数据端点，它提供了各种任务元数据和 Docker 统计信息添加到容器中。

任务元数据和联网速率统计数据将发送到 CloudWatch 容器洞察，并可在 AWS Management Console 中查看。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。

> **Note**
>
> Amazon ECS 提供较早版本的任务元数据端点。为避免将来需要创建新的任务元数据终端节点版本，可能会将其他元数据添加到版本 4 输出中。我们不会删除任何现有元数据或更改元数据字段名称。

原定设置情况下，环境变量会注入到运行至少版本的 Amazon EC2 Linux 实例上启动的 Amazon ECS 任务的容器中 1.39.0 的 Amazon ECS 容器代理。对于使用 awsvpc 网络模式的 Amazon EC2 Windows 实例，Amazon ECS 容器代理的版本必须至少为 1.54.0。有关更多信息，请参阅 Amazon ECS Linux 容器实例管理。

> **Note**
>
> 通过将代理更新为最新版本，您可以使用较旧版本的 Amazon ECS 容器代理在 Amazon EC2 实例上添加对此功能的支持。有关更多信息，请参阅 更新 Amazon ECS 容器代理。

任务元数据端点版本 4 路径

以下任务元数据端点路径可用于容器：

`${ECS_CONTAINER_METADATA_URI_V4}`

此路径返回容器的元数据。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路径返回任务的元数据，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Amazon ECS 任务元数据 V4 JSON 响应。

`${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`

除了可以使用 `ListTagsForResource` API 检索的任务和容器实例标记外，此路径还返回 `/task` 端点中包含的任务的元数据。检索标记元数据时收到的任何错误都将包含在 `Errors` 字段中的值。

> **Note**
>
> `Errors` 字段仅位于运行至少一个版本 `1.50.0` 的容器代理的 Amazon EC2 Linux 实例上托管的任务的响应中。对于使用 `awsvpc` 网络模式的 Amazon EC2 Windows 实例，Amazon ECS 容器代理必须至少为版本 `1.54.0`
> 这端点需要 `ecs.ListTagsForResource` 权限。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路径返回特定容器的 Docker 统计信息。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。

对于使用 `awsvpc` 或 `bridge` 版本托管在运行至少版本 `1.43.0` 的容器代理的Amazon EC2 Linux 实例上的或网络模式的Amazon ECS任务，响应中将包含其他网络速率统计信息。对于所有其他任务，响应将仅包含累积网络统计信息。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路径返回与任务相关的所有容器的 Docker 统计数据 。附加容器可以使用此路径提取网络指标。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。

对于使用 `awsvpc` 或 `bridge` 版本托管在运行至少版本 `1.43.0` 的容器代理的Amazon EC2 Linux实例上的或网络模式的Amazon ECS任务，响应中将包含其他网络速率统计信息。对于所有其他任务，响应将仅包含累积网络统计信息。

Amazon ECS 任务元数据 V4 JSON 响应

以下信息返回自任务元数据终端节点 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 响应。除了任务中每个容器的元数据外，还包括与任务关联的元数据。

`Cluster`

任务所属的 Amazon ECS 群集的 Amazon Resource Name (ARN) 或短名称。

`ServiceName`

任务所属服务的名称。如果任务与服务相关联，则 Amazon EC2 和 Amazon ECS Anywhere 容器实例将显示 ServiceName。

> **Note**
>
> 仅在使用 Amazon ECS 容器代理版本 `1.63.1` 或更高版本时包含 `ServiceName` 元数据。

`VPCID`

Amazon EC2 容器实例的 VPC ID。此字段仅针对 Amazon EC2 实例显示。

> **Note**
>
> 仅在使用 Amazon ECS 容器代理版本 `1.63.1` 或更高版本时包含 `VPCID` 元数据。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`Family`

任务的 Amazon ECS 任务定义系列。

`Revision`

任务的 Amazon ECS 任务定义修订。

`DesiredStatus`

来自 Amazon ECS 的任务的所需状态。

`KnownStatus`

来自 Amazon ECS 的任务的已知状态。

`Limits`

在任务级别上指定的资源限制，如 CPU（以 vCPU 表示）和内存。如果未定义资源限制，则省略此参数。

`PullStartedAt`

开始提取第一个容器映像时的时间戳。

`PullStoppedAt`

完成提取最后一个容器映像时的时间戳。

`AvailabilityZone`

任务所在的可用区。

> **Note**
>
> 可用区元数据仅适用于使用平台版本 1.4 或更高版本（Linux）或者 1.0.0（Windows）的 Fargate 任务。

`LaunchType`

任务使用的启动类型。使用集群容量提供程序时，这表明任务使用的是 Fargate 还是 EC2 基础设施。

> **Note**
>
> 仅在使用 Amazon ECS Linux 容器代理版本 1.45.0 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）时将该 LaunchType 元数据包含在内。

`Containers`

与任务关联的每个容器的容器元数据列表。

`DockerId`

容器的 Docker ID。

当您使用 Fargate 时，id 是一个 32 位十六进制，后面是 10 位数字。

`Name`

任务定义中所指定的容器的名称。

`DockerName`

提供给 Docker 的容器的名称。Amazon ECS 容器代理为容器生成一个唯一名称，以避免相同任务定义的多个副本在一个实例上运行时发生名称冲突。

`Image`

容器的映像。

`ImageID`

容器的 SHA-256 摘要。

`Ports`

对于容器公开的任何端口。如果没有公开的端口，则省略此参数。

`Labels`

应用到容器的任何标签。如果没有应用的标签，则省略此参数。

`DesiredStatus`

来自 Amazon ECS 的容器的所需状态。

`KnownStatus`

来自 Amazon ECS 的容器的已知状态。

`ExitCode`

容器的退出代码。如果没有容器退出，则省略此参数。

`Limits`

在容器级别上指定的资源限制，如 CPU（以 CPU 单位表示）和内存。如果未定义资源限制，则省略此参数。

`CreatedAt`

创建容器时的时间戳。如果尚未创建容器，则省略此参数。

`StartedAt`

容器启动时的时间戳。如果尚未启动容器，则省略此参数。

`FinishedAt`

容器停止时的时间戳。如果尚未停止容器，则省略此参数。

`Type`

容器的类型。在您的任务定义中指定的容器属于 `NORMAL` 类型。您可以省略其他被 Amazon ECS 容器代理用来进行内部任务资源预配置的容器类型。

`LogDriver`

容器使用的日志驱动程序。

> **Note**
>
> 该 `LogDriver` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`LogOptions`

为容器定义的日志驱动程序选项。

> **Note**
>
> 该 `LogOptions` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 该 `ContainerARN` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`Networks`

容器的网络信息，如网络模式和 IP 地址。如果未定义网络信息，则省略此参数。

`ExecutionStoppedAt`

任务的 `DesiredStatus` 变为 `STOPPED` 时的时间戳。这将发生在关键容器变成 `STOPPED` 时。

Amazon ECS 任务元数据 v4 示例

以下示例显示了每个任务元数据端点的输出示例。

容器元数据响应示例

查询 `${ECS_CONTAINER_METADATA_URI_V4}` 终端节点时，仅返回有关容器本身的元数据。下面是一个示例输出。

```
{
    "DockerId": "ea32192c8553fbff06c9340478a2ff089b2bb5646fb718b4ee206641c9086d66",
    "Name": "curl",
    "DockerName": "ecs-curltest-24-curl-cca48e8dcadd97805600",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/8f03e41243824aea923aca126495f665",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "24"
    },
    "DesiredStatus": "RUNNING",
```

```
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-02T00:15:07.620912337Z",
    "StartedAt": "2020-10-02T00:15:08.062559351Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/metadata",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/8f03e41243824aea923aca126495f665"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/0206b271-
b33f-47ab-86c6-a0ba208a70a9",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.100"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0e:9e:32:c7:48:85",
            "IPv4SubnetCIDRBlock": "10.0.2.0/24",
            "PrivateDNSName": "ip-10-0-2-100.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "10.0.2.1/24"
        }
    ]
}
```


任务元数据响应示例

查询 ${ECS_CONTAINER_METADATA_URI_V4}/task 端点时，除了任务中每个容器的元数据外，还会返回有关该容器的任务元数据。下面是一个示例输出。


```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
```

```
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "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-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
```

```
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
```

```
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

带标签元数据响应的示例任务

查询 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端点时，将返回有关任务的元数据，包括任务和容器实例标记。下面是一个示例输出。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "TaskTags": {
        "tag-use": "task-metadata-endpoint-test"
    },
    "ContainerInstanceTags":{
        "tag_key":"tag_value"
    },
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "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-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
```

```
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

带有错误元数据响应标签的示例任务

查询 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端点时，将返回有关任务的元数据，包括任务和容器实例标记。如果检索标记数据时出错，则响应中返回错误。以下是与容器实例关联的IAM角色没有允许的 ecs:ListTagsForResource 权限时的输出示例。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "Errors": [
        {
            "ErrorField": "ContainerInstanceTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:container-
instance/default/2dd1b186f39845a584488d2ef155c131",
            "StatusCode": 400,
            "RequestId": "cd597ef0-272b-4643-9bd2-1de469870fa6",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
        },
        {
            "ErrorField": "TaskTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3",
            "StatusCode": 400,
            "RequestId": "862c5986-6cd2-4aa6-87cc-70be395531e1",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3"
        }
    ],
    "LaunchType": "EC2",
    "Containers": [
        {
```

```
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "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-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
```

```
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
```

```
}
```

容器统计响应示例

查询 ${ECS_CONTAINER_METADATA_URI_V4}/stats 端点时，将返回容器的网络指标。对于使用 awsvpc 或 bridge 版本托管在运行至少版本 1.43.0 的容器代理的Amazon EC2实例上的或网络模式的Amazon ECS任务，响应中将包含其他网络速率统计信息。对于所有其他任务，响应将仅包含累积网络统计信息。

以下是 Amazon EC2 上使用 bridge 网络模式的 Amazon ECS 任务的输出示例。

```
{
    "read": "2020-10-02T00:51:13.410254284Z",
    "preread": "2020-10-02T00:51:12.406202398Z",
    "pids_stats": {
        "current": 3
    },
    "blkio_stats": {
        "io_service_bytes_recursive": [

        ],
        "io_serviced_recursive": [

        ],
        "io_queue_recursive": [

        ],
        "io_service_time_recursive": [

        ],
        "io_wait_time_recursive": [

        ],
        "io_merged_recursive": [

        ],
        "io_time_recursive": [

        ],
        "sectors_recursive": [

        ]
    },
```

```
    "num_procs": 0,
    "storage_stats": {

    },
    "cpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13939680000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "precpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13937670000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "memory_stats": {
        "usage": 1806336,
        "max_usage": 6299648,
        "stats": {
```

```
            "active_anon": 606208,
            "active_file": 0,
            "cache": 0,
            "dirty": 0,
            "hierarchical_memory_limit": 134217728,
            "hierarchical_memsw_limit": 268435456,
            "inactive_anon": 0,
            "inactive_file": 0,
            "mapped_file": 0,
            "pgfault": 4185,
            "pgmajfault": 0,
            "pgpgin": 2926,
            "pgpgout": 2778,
            "rss": 606208,
            "rss_huge": 0,
            "total_active_anon": 606208,
            "total_active_file": 0,
            "total_cache": 0,
            "total_dirty": 0,
            "total_inactive_anon": 0,
            "total_inactive_file": 0,
            "total_mapped_file": 0,
            "total_pgfault": 4185,
            "total_pgmajfault": 0,
            "total_pgpgin": 2926,
            "total_pgpgout": 2778,
            "total_rss": 606208,
            "total_rss_huge": 0,
            "total_unevictable": 0,
            "total_writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "limit": 134217728
    },
    "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
    "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
    "networks": {
        "eth0": {
            "rx_bytes": 84,
            "rx_packets": 2,
            "rx_errors": 0,
            "rx_dropped": 0,
            "tx_bytes": 84,
```

```
            "tx_packets": 2,
            "tx_errors": 0,
            "tx_dropped": 0
        }
    },
    "network_rate_stats": {
        "rx_bytes_per_sec": 0,
        "tx_bytes_per_sec": 0
    }
}
```

示例任务统计信息响应

查询 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 终端节点时，将返回有关容器所属的任务的网络指标。下面是一个示例输出。

```
{
    "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854": {
        "read": "2020-10-02T00:51:32.51467703Z",
        "preread": "2020-10-02T00:51:31.50860463Z",
        "pids_stats": {
            "current": 1
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [
```

```
            ],
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13977820000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
```

```
            }
        },
        "memory_stats": {
            "usage": 532480,
            "max_usage": 6279168,
            "stats": {
                "active_anon": 40960,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 9223372036854771712,
                "hierarchical_memsw_limit": 9223372036854771712,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 2033,
                "pgmajfault": 0,
                "pgpgin": 1734,
                "pgpgout": 1724,
                "rss": 40960,
                "rss_huge": 0,
                "total_active_anon": 40960,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 2033,
                "total_pgmajfault": 0,
                "total_pgpgin": 1734,
                "total_pgpgout": 1724,
                "total_rss": 40960,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 4073377792
        },
        "name": "/ecs-curltest-26-internalecspause-a6bcc3dbadfacfe85300",
        "id": "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854",
        "networks": {
```

```
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    },
    "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af": {
        "read": "2020-10-02T00:51:32.512771349Z",
        "preread": "2020-10-02T00:51:31.510597736Z",
        "pids_stats": {
            "current": 3
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
```

```
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 379075681,
                "percpu_usage": [
                    191355275,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13977800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 378825197,
                "percpu_usage": [
                    191104791,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
```

```
        "memory_stats": {
            "usage": 1814528,
            "max_usage": 6299648,
            "stats": {
                "active_anon": 606208,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 134217728,
                "hierarchical_memsw_limit": 268435456,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 5377,
                "pgmajfault": 0,
                "pgpgin": 3613,
                "pgpgout": 3465,
                "rss": 606208,
                "rss_huge": 0,
                "total_active_anon": 606208,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 5377,
                "total_pgmajfault": 0,
                "total_pgpgin": 3613,
                "total_pgpgout": 3465,
                "total_rss": 606208,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 134217728
        },
        "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
        "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
```

```
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    }
}
```

Amazon ECS 任务元数据端点版本 3

> **Important**
>
> 任务元数据版本 3 端点不再主动维护。我们建议您更新任务元数据版本 4 端点以获取最新的元数据端点信息。有关更多信息，请参阅 the section called “任务元数据终端节点版本 4”。
> 如果您使用的是托管在 AWS Fargate 上的 Amazon ECS 任务，请参阅 AWS FargateAmazon Elastic Container Service 用户指南中的任务元数据端点版本 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。对于使用 `bridge` 网络模式的任务，查询 `/stats` 终端节点时可以使用网络指标。

预设情况下，对于在平台版本v1.3.0或更高版本上使用 Fargate 启动类型的任务，以及使用 EC2 启动类型并在运行至少版本 1.21.0 的 Amazon ECS 容器代理的 Amazon EC2 Linux 基础设施或在 Amazon EC2 Windows 基础设施上启动的任务，会启用任务元数据端点版本3功能运行至少版本 `1.54.0` 的 Amazon ECS 容器代理并使用 `awsvpc` 网络模式。有关更多信息，请参阅 Amazon ECS Linux 容器实例管理。

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。

> **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 端点。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 2。

任务元数据端点版本 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Amazon ECS 任务元数据 v3 JSON 响应。

`${ECS_CONTAINER_METADATA_URI}/taskWithTags`

除了可以使用 `ListTagsForResource` API 检索的任务和容器实例标记外，此路径还返回 `/task` 端点中包含的任务的元数据。

`${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。

Amazon ECS 任务元数据 v3 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`

任务所属的 Amazon ECS 群集的 Amazon Resource Name (ARN) 或短名称。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`Family`

任务的 Amazon ECS 任务定义系列。

`Revision`

任务的 Amazon ECS 任务定义修订。

`DesiredStatus`

来自 Amazon ECS 的任务的所需状态。

`KnownStatus`

来自 Amazon ECS 的任务的已知状态。

`Limits`

在任务级别上指定的资源限制，如 CPU（以 vCPU 表示）和内存。如果未定义资源限制，则省略此参数。

`PullStartedAt`

开始提取第一个容器映像时的时间戳。

`PullStoppedAt`

完成提取最后一个容器映像时的时间戳。

`AvailabilityZone`

任务所在的可用区。

> **Note**
>
> 可用区元数据仅适用于使用平台版本 1.4 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）的 Fargate 任务。

`Containers`

与任务关联的每个容器的容器元数据列表。

`DockerId`

容器的 Docker ID。

`Name`

任务定义中所指定的容器的名称。

`DockerName`

提供给 Docker 的容器的名称。Amazon ECS 容器代理为容器生成一个唯一名称，以避免相同任务定义的多个副本在一个实例上运行时发生名称冲突。

`Image`

容器的映像。

`ImageID`

容器的 SHA-256 摘要。

`Ports`

对于容器公开的任何端口。如果没有公开的端口，则省略此参数。

`Labels`

应用到容器的任何标签。如果没有应用的标签，则省略此参数。

`DesiredStatus`

来自 Amazon ECS 的容器的所需状态。

`KnownStatus`

来自 Amazon ECS 的容器的已知状态。

`ExitCode`

容器的退出代码。如果没有容器退出，则省略此参数。

`Limits`

在容器级别上指定的资源限制，如 CPU（以 CPU 单位表示）和内存。如果未定义资源限制，则省略此参数。

`CreatedAt`

创建容器时的时间戳。如果尚未创建容器，则省略此参数。

`StartedAt`

容器启动时的时间戳。如果尚未启动容器，则省略此参数。

FinishedAt

容器停止时的时间戳。如果尚未停止容器，则省略此参数。

Type

容器的类型。在您的任务定义中指定的容器属于 NORMAL 类型。您可以省略其他被 Amazon ECS 容器代理用来进行内部任务资源预配置的容器类型。

Networks

容器的网络信息，如网络模式和 IP 地址。如果未定义网络信息，则省略此参数。

ClockDrift

有关参考时间和系统时间之间差异的信息。这适用于 Linux 操作系统。此功能使用 Amazon Time Sync Service 来测量时钟精度，并提供容器绑定的时钟误差。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的为您的 Linux 实例设定时间。

ReferenceTime

时钟准确度的基础。Amazon ECS 通过 NTP 使用协调世界时（UTC）全球标准，例如 2021-09-07T16:57:44Z。

ClockErrorBound

时钟误差的度量，定义为与 UTC 的偏移量。此错误是参考时间和系统时间之间的差异（以毫秒为单位）。

ClockSynchronizationStatus

指示系统时间和参考时间之间的最近一次同步尝试是否成功。

有效值为 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

ExecutionStoppedAt

任务的 DesiredStatus 变为 STOPPED 时的时间戳。这将发生在关键容器变成 STOPPED 时。

Amazon ECS 任务元数据 v3 示例

以下示例显示来自任务元数据终端节点的示例输出。

容器元数据响应示例

查询 ${ECS_CONTAINER_METADATA_URI} 终端节点时，仅返回有关容器本身的元数据。下面是一个示例输出。

```
{
    "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"
            ]
        }
    ]
}
```

任务元数据响应示例

查询 ${ECS_CONTAINER_METADATA_URI}/task 终端节点时，将返回有关容器所属的任务的元数据。下面是一个示例输出。

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 任务元数据端点版本 2

> **Important**
>
> 任务元数据版本 2 端点不再主动维护。我们建议您更新任务元数据版本 4 端点以获取最新的元数据端点信息。有关更多信息，请参阅 the section called “任务元数据终端节点版本 4”。

从 1.17.0 版容器代理开始，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 端点上使用 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端点时，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 适用于 Amazon ECS 的 Fargate Linux 平台版本。
- 使用 EC2 启动类型的任务也使用 awsvpc 网络模式，并且在运行至少版本 1.17.0 的 Amazon ECS 容器代理的 Amazon EC2 Linux 基础设施或运行至少版本 1.54.0 的 Amazon ECS 容器代理的 Amazon EC2 Windows 基础设施上启动。有关更多信息，请参阅 Amazon ECS Linux 容器实例管理。

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。

任务元数据端点路径

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。

`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/metadata/taskWithTags`

除了可以使用 ListTagsForResource API 检索的任务和容器实例标记外，此路径还返回 /task 端点中包含的任务的元数据。

`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`

任务所属的 Amazon ECS 群集的 Amazon Resource Name (ARN) 或短名称。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`Family`

任务的 Amazon ECS 任务定义系列。

`Revision`

任务的 Amazon ECS 任务定义修订。

`DesiredStatus`

来自 Amazon ECS 的任务的所需状态。

`KnownStatus`

来自 Amazon ECS 的任务的已知状态。

`Limits`

在任务级别上指定的资源限制，如 CPU（以 vCPU 表示）和内存。如果未定义资源限制，则省略此参数。

`PullStartedAt`

开始提取第一个容器映像时的时间戳。

`PullStoppedAt`

完成提取最后一个容器映像时的时间戳。

`AvailabilityZone`

任务所在的可用区。

> **Note**
>
> 可用区元数据仅适用于使用平台版本 1.4 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）的 Fargate 任务。

`Containers`

与任务关联的每个容器的容器元数据列表。

`DockerId`

容器的 Docker ID。

`Name`

任务定义中所指定的容器的名称。

`DockerName`

提供给 Docker 的容器的名称。Amazon ECS 容器代理为容器生成一个唯一名称，以避免相同任务定义的多个副本在一个实例上运行时发生名称冲突。

`Image`

容器的映像。

`ImageID`

容器的 SHA-256 摘要。

`Ports`

对于容器公开的任何端口。如果没有公开的端口，则省略此参数。

`Labels`

应用到容器的任何标签。如果没有应用的标签，则省略此参数。

`DesiredStatus`

来自 Amazon ECS 的容器的所需状态。

`KnownStatus`

来自 Amazon ECS 的容器的已知状态。

`ExitCode`

容器的退出代码。如果没有容器退出，则省略此参数。

`Limits`

在容器级别上指定的资源限制，如 CPU（以 CPU 单位表示）和内存。如果未定义资源限制，则省略此参数。

`CreatedAt`

创建容器时的时间戳。如果尚未创建容器，则省略此参数。

`StartedAt`

容器启动时的时间戳。如果尚未启动容器，则省略此参数。

`FinishedAt`

容器停止时的时间戳。如果尚未停止容器，则省略此参数。

`Type`

容器的类型。在您的任务定义中指定的容器属于 `NORMAL` 类型。您可以省略其他被 Amazon ECS 容器代理用来进行内部任务资源预配置的容器类型。

`Networks`

容器的网络信息，如网络模式和 IP 地址。如果未定义网络信息，则省略此参数。

`ClockDrift`

有关参考时间和系统时间之间差异的信息。这适用于 Linux 操作系统。此功能使用 Amazon Time Sync Service 来测量时钟精度，并提供容器绑定的时钟误差。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的为您的 Linux 实例设定时间。

`ReferenceTime`

时钟准确度的基础。Amazon ECS 通过 NTP 使用协调世界时（UTC）全球标准，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

时钟误差的度量，定义为与 UTC 的偏移量。此错误是参考时间和系统时间之间的差异（以毫秒为单位）。

`ClockSynchronizationStatus`

指示系统时间和参考时间之间的最近一次同步尝试是否成功。

有效值为 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

ExecutionStoppedAt

任务的 DesiredStatus 变为 STOPPED 时的时间戳。这将发生在关键容器变成 STOPPED 时。

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 任务元数据可用于 Fargate 上的任务

Fargate 上的 Amazon ECS 提供了检索有关任务和容器的各种任务元数据、网络指标和 Docker 统计数据的方法。这称为任务元数据端点。以下任务元数据端点版本适用于 Fargate 任务上的 Amazon ECS：

- 任务元数据端点版本 4 - 适用于在平台版本 1.4.0 或更高版本上的任务。
- 任务元数据端点版本 3 - 适用于在平台版本 1.1.0 或更高版本上的任务。

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，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。

主题

- Fargate 上任务的 Amazon ECS 任务元数据端点版本 4
- Fargate 上任务的 Amazon ECS 任务元数据端点版本 3

Fargate 上任务的 Amazon ECS 任务元数据端点版本 4

> **Important**
>
> 如果您正在使用托管在 Amazon EC2 实例上的 Amazon ECS 任务，请参阅 Amazon ECS 任务元数据端点。

从 Fargate 平台版本 1.4.0 开始，名为 ECS_CONTAINER_METADATA_URI_V4 的环境变量被注入到任务中的每个容器中。在您查询任务元数据版本 4 端点时，将为任务提供各种任务元数据和 Docker 统计数据。

任务元数据版本 4 端点的功能与版本 3 端点相似，但为容器和任务提供了额外的网络元数据。查询 /stats 终端节点时还可以使用其他网络指标。

默认情况下，对于使用平台版本 1.4.0 或更高版本在 AWS Fargate 上运行的所有 Amazon ECS 任务，都会启用任务元数据端点。

> **Note**
>
> 为避免将来需要创建新的任务元数据终端节点版本，可能会将其他元数据添加到版本 4 输出中。我们不会删除任何现有元数据或更改元数据字段名称。

Fargate 任务元数据端点版本 4 路径

以下任务元数据终端节点可用于容器：

`${ECS_CONTAINER_METADATA_URI_V4}`

此路径返回容器的元数据。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路径返回任务的元数据，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Fargate 上任务的 Amazon ECS 任务元数据 v4 JSON 响应。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路径返回 Docker 容器的 Docker 统计数据。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。

> **Note**
>
> AWS Fargate 上的 Amazon ECS 任务要求容器在返回容器统计数据之前运行约 1 秒。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路径返回与任务相关的所有容器的 Docker 统计数据 。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。

> **Note**
>
> AWS Fargate 上的 Amazon ECS 任务要求容器在返回容器统计数据之前运行约 1 秒。

Fargate 上任务的 Amazon ECS 任务元数据 v4 JSON 响应

以下元数据返回自任务元数据端点 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 响应。

`Cluster`

任务所属的 Amazon ECS 群集的 Amazon Resource Name (ARN) 或短名称。

`VPCID`

Amazon EC2 容器实例的 VPC ID。此字段仅针对 Amazon EC2 实例显示。

> **Note**
>
> 仅在使用 Amazon ECS 容器代理版本 `1.63.1` 或更高版本时包含 `VPCID` 元数据。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`Family`

任务的 Amazon ECS 任务定义系列。

`Revision`

任务的 Amazon ECS 任务定义修订。

`DesiredStatus`

来自 Amazon ECS 的任务的所需状态。

`KnownStatus`

来自 Amazon ECS 的任务的已知状态。

`Limits`

在任务级别上指定的资源限制，如 CPU（以 vCPU 表示）和内存。如果未定义资源限制，则省略此参数。

`PullStartedAt`

开始提取第一个容器映像时的时间戳。

`PullStoppedAt`

完成提取最后一个容器映像时的时间戳。

`AvailabilityZone`

任务所在的可用区。

> **Note**
>
> 可用区元数据仅适用于使用平台版本 1.4 或更高版本（Linux）或者 1.0.0（Windows）的 Fargate 任务。

`LaunchType`

任务使用的启动类型。使用集群容量提供程序时，这表明任务使用的是 Fargate 还是 EC2 基础设施。

> **Note**
>
> 仅在使用 Amazon ECS Linux 容器代理版本 `1.45.0` 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）时将该 `LaunchType` 元数据包含在内。

`EphemeralStorageMetrics`

此任务短暂存储的预留大小和当前使用情况。

> **Note**
>
> Fargate 可保留磁盘空间。该磁盘空间仅由 Fargate 使用。您无需为此付费。它没有显示在这些指标中。但是，您可以在 `df` 等其他工具中看到这种额外的存储空间。

`Utilized`

此任务的当前短暂存储使用量（MiB 为单位）。

`Reserved`

此任务的预留短暂存储（MiB 为单位）。无法在正在运行的任务中更改短暂存储大小。您可以在任务定义中指定的 `ephermalStorage` 对象以更改短暂存储量。以 GiB 而不是 MiB 为单位

指定 ephermalStorage。ephermalStorage 和 EphemeralStorageMetrics 仅适用于 Fargate Linux 平台 1.4.0 或更高版本上。

Containers

与任务关联的每个容器的容器元数据列表。

DockerId

容器的 Docker ID。

当您使用 Fargate 时，id 是一个 32 位十六进制，后面是 10 位数字。

Name

任务定义中所指定的容器的名称。

DockerName

提供给 Docker 的容器的名称。Amazon ECS 容器代理为容器生成一个唯一名称，以避免相同任务定义的多个副本在一个实例上运行时发生名称冲突。

Image

容器的映像。

ImageID

容器的 SHA-256 摘要。

Ports

对于容器公开的任何端口。如果没有公开的端口，则省略此参数。

Labels

应用到容器的任何标签。如果没有应用的标签，则省略此参数。

DesiredStatus

来自 Amazon ECS 的容器的所需状态。

KnownStatus

来自 Amazon ECS 的容器的已知状态。

ExitCode

容器的退出代码。如果没有容器退出，则省略此参数。

`Limits`

在容器级别上指定的资源限制，如 CPU（以 CPU 单位表示）和内存。如果未定义资源限制，则省略此参数。

`CreatedAt`

创建容器时的时间戳。如果尚未创建容器，则省略此参数。

`StartedAt`

容器启动时的时间戳。如果尚未启动容器，则省略此参数。

`FinishedAt`

容器停止时的时间戳。如果尚未停止容器，则省略此参数。

`Type`

容器的类型。在您的任务定义中指定的容器属于 `NORMAL` 类型。您可以省略其他被 Amazon ECS 容器代理用来进行内部任务资源预配置的容器类型。

`LogDriver`

容器使用的日志驱动程序。

> **Note**
>
> 该 `LogDriver` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`LogOptions`

为容器定义的日志驱动程序选项。

> **Note**
>
> 该 `LogOptions` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 该 `ContainerARN` 元数据仅在使用Amazon ECS Linux 容器代理版本时包含 `1.45.0` 或更高版本。

`Networks`

容器的网络信息，如网络模式和 IP 地址。如果未定义网络信息，则省略此参数。

`Snapshotter`

containerd 用于下载此容器映像的 snapshotter。有效值为 `overlayfs`（默认值）和延迟加载 SOCI 索引时使用的 `soci`。该参数仅适用于在 Linux 平台版本 `1.4.0` 上运行的任务。

`ClockDrift`

有关参考时间和系统时间之间差异的信息。此功能使用 Amazon Time Sync Service 来测量时钟精度，并提供容器绑定的时钟误差。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的为您的 Linux 实例设定时间。

`ReferenceTime`

时钟准确度的基础。Amazon ECS 通过 NTP 使用协调世界时（UTC）全球标准，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

时钟误差的度量，定义为与 UTC 的偏移量。此错误是参考时间和系统时间之间的差异（以毫秒为单位）。

`ClockSynchronizationStatus`

指示系统时间和参考时间之间的最近一次同步尝试是否成功。

有效值为 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

`ExecutionStoppedAt`

任务的 `DesiredStatus` 变为 `STOPPED` 时的时间戳。这将发生在关键容器变成 `STOPPED` 时。

Fargate 上任务的 Amazon ECS 任务元数据 v4 示例

以下示例显示了在上 AWS Fargate 运行的 Amazon ECS 任务的任务元数据端点的输出示例。

通过容器，您可以在任务元数据端点后面使用 curl 来查询端点，例如 `curl ${ECS_CONTAINER_METADATA_URI_V4}/task`。

容器元数据响应示例

查询 `${ECS_CONTAINER_METADATA_URI_V4}` 终端节点时，仅返回有关容器本身的元数据。下面是一个示例输出。

```
{
    "DockerId": "cd189a933e5849daa93386466019ab50-2495160603",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:25f3695bedfb454a50f12d127839a68ad3caf91e451c1da073db34c542c4d2cb",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/default/
cd189a933e5849daa93386466019ab50",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "2"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-08T20:09:11.44527186Z",
    "StartedAt": "2020-10-08T20:09:11.44527186Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "192.0.2.3"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0a:de:f6:10:51:e5",
            "IPv4SubnetCIDRBlock": "192.0.2.0/24",
            "DomainNameServers": [
```

```
                "192.0.2.2"
            ],
            "DomainNameSearchList": [
                "us-west-2.compute.internal"
            ],
            "PrivateDNSName": "ip-10-0-0-222.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "192.0.2.0/24"
        }
    ],
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/05966557-
f16c-49cb-9352-24b3a0dcd0e1",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/containerlogs",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/cd189a933e5849daa93386466019ab50"
    },
    "LogDriver": "awslogs",
    "Snapshotter": "overlayfs"
}
```

Fargate 上任务的 Amazon ECS 任务元数据 v4 示例

查询 ${ECS_CONTAINER_METADATA_URI_V4}/task 终端节点时，将返回有关容器所属的任务的元数据。下面是一个示例输出。

```
{
  "Cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/clusterName",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyEmptyCluster/
bfa2636268144d039771334145e490c5",
  "Family": "sample-fargate",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": {
    "CPU": 0.25,
    "Memory": 512
  },
  "PullStartedAt": "2023-07-21T15:45:33.532811081Z",
  "PullStoppedAt": "2023-07-21T15:45:38.541068435Z",
  "AvailabilityZone": "us-east-1d",
  "Containers": [
    {
```

```
      "DockerId": "bfa2636268144d039771334145e490c5-1117626119",
      "Name": "curl-image",
      "DockerName": "curl-image",
      "Image": "curlimages/curl",
      "ImageID":
 "sha256:daf3f46a2639c1613b25e85c9ee4193af8a1d538f92483d67f9a3d7f21721827",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "curl-image",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 128 },
      "CreatedAt": "2023-07-21T15:45:44.91368314Z",
      "StartedAt": "2023-07-21T15:45:44.91368314Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/da6cccf7-1178-400c-afdf-7536173ee209",
      "Snapshotter": "overlayfs"
    },
    {
      "DockerId": "bfa2636268144d039771334145e490c5-3681984407",
      "Name": "fargate-app",
      "DockerName": "fargate-app",
      "Image": "public.ecr.aws/docker/library/httpd:latest",
```

```
      "ImageID":
 "sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5ececa02",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "fargate-app",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 2 },
      "CreatedAt": "2023-07-21T15:45:44.954460255Z",
      "StartedAt": "2023-07-21T15:45:44.954460255Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/f65b461d-aa09-4acb-a579-9785c0530cbc",
      "Snapshotter": "overlayfs"
    }
  ],
  "LaunchType": "FARGATE",
  "ClockDrift": {
    "ClockErrorBound": 0.446931,
    "ReferenceTimestamp": "2023-07-21T16:09:17Z",
    "ClockSynchronizationStatus": "SYNCHRONIZED"
  },
  "EphemeralStorageMetrics": {
    "Utilized": 261,
    "Reserved": 20496
```

```
  }
}
```

示例任务统计信息响应

查询 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 终端节点时，将返回有关容器所属的任务的网络指标。下面是一个示例输出。

```
{
  "3d1f891cded94dc795608466cce8ddcf-464223573": {
    "read": "2020-10-08T21:24:44.938937019Z",
    "preread": "2020-10-08T21:24:34.938633969Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
```

```
          "value": 638976
        }
      ],
      "io_serviced_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 12
        }
      ],
      "io_queue_recursive": [],
      "io_service_time_recursive": [],
      "io_wait_time_recursive": [],
      "io_merged_recursive": [],
      "io_time_recursive": [],
      "sectors_recursive": []
    },
    "num_procs": 0,
    "storage_stats": {},
```

```
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 1137691504,
        "percpu_usage": [
          696479228,
          441212276,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9393210000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "precpu_stats": {
      "cpu_usage": {
        "total_usage": 1136624601,
        "percpu_usage": [
          695639662,
          440984939,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
```

```
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9373330000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "memory_stats": {
      "usage": 6504448,
      "max_usage": 8458240,
      "stats": {
        "active_anon": 1675264,
        "active_file": 557056,
        "cache": 651264,
        "dirty": 0,
        "hierarchical_memory_limit": 536870912,
        "hierarchical_memsw_limit": 9223372036854772000,
        "inactive_anon": 0,
        "inactive_file": 3088384,
        "mapped_file": 430080,
        "pgfault": 11034,
        "pgmajfault": 5,
        "pgpgin": 8436,
        "pgpgout": 7137,
        "rss": 4669440,
        "rss_huge": 0,
        "total_active_anon": 1675264,
        "total_active_file": 557056,
        "total_cache": 651264,
        "total_dirty": 0,
        "total_inactive_anon": 0,
        "total_inactive_file": 3088384,
        "total_mapped_file": 430080,
```

```
        "total_pgfault": 11034,
        "total_pgmajfault": 5,
        "total_pgpgin": 8436,
        "total_pgpgout": 7137,
        "total_rss": 4669440,
        "total_rss_huge": 0,
        "total_unevictable": 0,
        "total_writeback": 0,
        "unevictable": 0,
        "writeback": 0
      },
      "limit": 9223372036854772000
    },
    "name": "curltest",
    "id": "3d1f891cded94dc795608466cce8ddcf-464223573",
    "networks": {
      "eth1": {
        "rx_bytes": 2398415937,
        "rx_packets": 1898631,
        "rx_errors": 0,
        "rx_dropped": 0,
        "tx_bytes": 1259037719,
        "tx_packets": 428002,
        "tx_errors": 0,
        "tx_dropped": 0
      }
    },
    "network_rate_stats": {
      "rx_bytes_per_sec": 43.298687872232854,
      "tx_bytes_per_sec": 215.39347269466413
    }
  }
}
```


Fargate 上任务的 Amazon ECS 任务元数据端点版本 3

> **Important**
>
> 任务元数据版本 3 端点不再主动维护。我们建议您更新任务元数据版本 4 端点以获取最新的元数据端点信息。有关更多信息，请参阅 the section called “Fargate 上任务的任务元数据端点版本 4”。

从 Fargate 平台版本 1.1.0 开始，名为 ECS_CONTAINER_METADATA_URI 的环境变量被注入到任务中的每个容器中。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。

预设情况下，Fargate 上托管的使用平台版本 1.1.0 或更高版本的 Amazon ECS 任务将启用任务元数据端点功能。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。

Fargate 上任务的任务元数据端点路径

以下 API 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Fargate 上任务的 Amazon ECS 任务元数据 v3 JSON 响应。

`${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。

Fargate 上任务的 Amazon ECS 任务元数据 v3 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`

任务所属的 Amazon ECS 群集的 Amazon Resource Name (ARN) 或短名称。

`TaskARN`

容器所属的任务的完整 Amazon Resource Name (ARN)。

`Family`

任务的 Amazon ECS 任务定义系列。

`Revision`

任务的 Amazon ECS 任务定义修订。

`DesiredStatus`

来自 Amazon ECS 的任务的所需状态。

`KnownStatus`

来自 Amazon ECS 的任务的已知状态。

`Limits`

在任务级别上指定的资源限制，如 CPU（以 vCPU 表示）和内存。如果未定义资源限制，则省略此参数。

`PullStartedAt`

开始提取第一个容器映像时的时间戳。

`PullStoppedAt`

完成提取最后一个容器映像时的时间戳。

`AvailabilityZone`

任务所在的可用区。

> **Note**
>
> 可用区元数据仅适用于使用平台版本 1.4 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）的 Fargate 任务。

`Containers`

与任务关联的每个容器的容器元数据列表。

`DockerId`

容器的 Docker ID。

`Name`

任务定义中所指定的容器的名称。

`DockerName`

提供给 Docker 的容器的名称。Amazon ECS 容器代理为容器生成一个唯一名称，以避免相同任务定义的多个副本在一个实例上运行时发生名称冲突。

`Image`

容器的映像。

`ImageID`

容器的 SHA-256 摘要。

`Ports`

对于容器公开的任何端口。如果没有公开的端口，则省略此参数。

`Labels`

应用到容器的任何标签。如果没有应用的标签，则省略此参数。

`DesiredStatus`

来自 Amazon ECS 的容器的所需状态。

`KnownStatus`

来自 Amazon ECS 的容器的已知状态。

`ExitCode`

容器的退出代码。如果没有容器退出，则省略此参数。

`Limits`

在容器级别上指定的资源限制，如 CPU（以 CPU 单位表示）和内存。如果未定义资源限制，则省略此参数。

`CreatedAt`

创建容器时的时间戳。如果尚未创建容器，则省略此参数。

`StartedAt`

容器启动时的时间戳。如果尚未启动容器，则省略此参数。

`FinishedAt`

容器停止时的时间戳。如果尚未停止容器，则省略此参数。

Type

容器的类型。在您的任务定义中指定的容器属于 NORMAL 类型。您可以省略其他被 Amazon ECS 容器代理用来进行内部任务资源预配置的容器类型。

Networks

容器的网络信息，如网络模式和 IP 地址。如果未定义网络信息，则省略此参数。

ClockDrift

有关参考时间和系统时间之间差异的信息。这适用于 Linux 操作系统。此功能使用 Amazon Time Sync Service 来测量时钟精度，并提供容器绑定的时钟误差。有关更多信息，请参阅《适用于 Linux 实例的 Amazon EC2 用户指南》中的为您的 Linux 实例设定时间。

ReferenceTime

时钟准确度的基础。Amazon ECS 通过 NTP 使用协调世界时（UTC）全球标准，例如 2021-09-07T16:57:44Z。

ClockErrorBound

时钟误差的度量，定义为与 UTC 的偏移量。此错误是参考时间和系统时间之间的差异（以毫秒为单位）。

ClockSynchronizationStatus

指示系统时间和参考时间之间的最近一次同步尝试是否成功。

有效值为 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

ExecutionStoppedAt

任务的 DesiredStatus 变为 STOPPED 时的时间戳。这将发生在关键容器变成 STOPPED 时。

Fargate 上任务的 Amazon ECS 任务元数据 v3 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。

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。

```
curl -s http://localhost:51678/v1/metadata | python3 -mjson.tool
```

输出：

```
{
    "Cluster": "cluster_name",
    "ContainerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/cluster_name/container_instance_id",
    "Version": "Amazon ECS Agent - v1.30.0 (02ff320c)"
}
```

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：

```
curl http://localhost:51678/v1/tasks
```

输出：

```
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/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",
          "CreatedAt": "2023-10-08T20:09:11.44527186Z",
          "StartedAt": "2023-10-08T20:09:11.44527186Z",
          "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        },
        {
```

```
            "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。

```
curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

输出：

```
{
    "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId":
 "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx",
            "CreatedAt": "2023-10-08T20:09:11.44527186Z",
```

```
            "StartedAt": "2023-10-08T20:09:11.44527186Z",
            "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

使用运行时监控识别未经授权的行为

Amazon GuardDuty 是一项威胁检测服务，可帮助保护您的账户、容器、工作负载和 AWS 环境中的数据。GuardDuty 使用机器学习（ML）模型以及异常和威胁检测功能，持续监控不同的日志源和运行时活动，以识别环境中的潜在安全风险和恶意活动并确定其优先级。

GuardDuty 中的运行时监控通过持续监控 AWS 日志和联网活动来识别恶意或未经授权的行为，从而保护在 Fargate 和 EC2 容器实例上运行的工作负载。运行时监控使用轻量级、完全托管的 GuardDuty 安全代理分析主机中的行为，例如文件访问、进程执行和网络连接。它涉及的问题包括权限升级、使用公开的凭证，或与恶意 IP 地址、域通信，以及您的 Amazon EC2 实例和容器工作负载上存在恶意软件。有关更多信息，请参阅《GuardDuty User Guide》中的 GuardDuty Runtime Monitoring。

您的安全管理员为 AWS Organizations 中的单个或多个账户启用 GuardDuty 的运行时监控。他们还会选择在您使用 Fargate 时 GuardDuty 是否自动部署 GuardDuty 安全代理。您的所有集群都将自动受到保护，GuardDuty 将代表您管理安全代理。

在以下情况下，您还可以手动配置 GuardDuty 安全代理：

- 您使用 EC2 容器实例时
- 您需要精细控制在集群级别启用运行时监控时

要使用运行时监控，您必须配置受保护的集群，并在您的 EC2 容器实例上安装和管理 GuardDuty 安全代理。

运行时监控如何与 Amazon ECS 结合使用

运行时监控使用轻量级 GuardDuty 安全代理，以监控 Amazon ECS 的工作负载活动，了解应用程序请求、访问和消耗底层系统资源的方式。

对于 Fargate 任务，GuardDuty 安全代理作为每项任务的附加容器运行。

对于 EC2 容器实例，GuardDuty 安全代理作为进程在实例上运行。

GuardDuty 安全代理从以下资源收集数据，然后将数据发送到 GuardDuty 以进行处理。您可以在 GuardDuty 控制台中查看调查发现。您也可以将它们发送给其他 AWS 服务（例如 AWS Security Hub）或第三方安全供应商，以进行聚合和修复。有关如何查看和管理调查发现的信息，请参阅《Amazon GuardDuty User Guide》中的 Managing Amazon GuardDuty findings。

- 来自以下 Amazon ECS API 调用的响应：
  - DescribeClusters

    使用 `--include TAGS` 选项时，响应参数包括运行时监控标签（设置标签时）。
  - DescribeTasks

    对于 Fargate 启动类型，响应参数包括 GuardDuty 附加容器。
  - ListAccountSettings

    响应参数包括运行时监控账户设置，该设置由您的安全管理员设置。
- 容器代理自检数据。有关更多信息，请参阅 Amazon ECS 容器自检。
- 启动类型的任务元数据端点：
  - Amazon ECS 任务元数据端点版本 4
  - Fargate 上任务的 Amazon ECS 任务元数据端点版本 4

注意事项

使用运行时监控时，请考虑以下内容：

- 运行时监控具有与之相关的成本。有关更多信息，请参阅 Amazon GuardDuty 定价。
- Amazon ECS Anywhere 不支持运行时监控。
- Windows 操作系统不支持运行时监控。
- 在 Fargate 上使用 Amazon ECS Exec 时，您必须指定容器名称，因为 GuardDuty 安全代理作为附加容器运行。
- 您不能在 GuardDuty 安全代理附加容器上使用 Amazon ECS Exec。
- 在集群级别控制运行时监控的 IAM 用户，必须具有适当的 IAM 权限才能进行标记。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。

- Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理存储在 Amazon ECR 私有存储库中的 GuardDuty 安全代理的权限。

资源利用率

您添加到集群的标签将计入集群标签配额。

GuardDuty 代理附加容器不计入每个任务定义的容器配额。

与大多数安全软件一样，GuardDuty 有少量开销。有关 Fargate 内存限制的信息，请参阅《GuardDuty User Guide》中的 CPU and memory limits。有关 Amazon EC2 内存限制的信息，请参阅 CPU and memory limit for GuardDuty agent。

Amazon ECS Fargate 工作负载的运行时监控

如果您使用 EC2 容器实例，则必须手动配置运行时监控。有关更多信息，请参阅 Amazon ECS 上的 EC2 工作负载的运行时监控。

您可以让 GuardDuty 管理您的容器实例上的安全代理。此选项仅可用于 Fargate。此选项（GuardDuty 代理管理）在 GuardDuty 中可用

当使用 GuardDuty 代理管理时，GuardDuty 会执行以下操作：

- 为托管集群的每个 VPC 创建 GuardDuty 的 VPC 端点。
- 检索最新的 GuardDuty 安全代理，并将其作为附加容器安装到所有新的独立 Fargate 任务和新的服务部署中。

  新的服务部署发生在您首次启动服务时，或者在您使用强制执行新部署选项更新现有服务时。

为 Amazon ECS 开启运行时监控

您可以配置 GuardDuty，以自动管理所有 Fargate 集群的安全代理。

以下是使用运行时监控的先决条件：

- Fargate 平台版本必须为适用于 Linux 的 `1.4.0` 或更高版本。
- Amazon ECS 的 IAM 角色和权限：
  - Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理 GuardDuty 安全代理的权限。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色。

  - 您可以使用预定义的标签来控制集群的运行时监控。如果您的访问策略基于标签限制访问，则必须向您的 IAM 用户授予标记集群的明确权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。
- 连接到 Amazon ECR 存储库：

  GuardDuty 安全代理存储于 Amazon ECR 存储库中。每个独立任务和服务任务都必须具有访问该存储库的权限。可以使用以下选项之一：
  - 对于公有子网中的任务，您可以对任务使用公有 IP 地址，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。
  - 对于私有子网中的任务，您可以使用网络地址转换（NAT）网关，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。

    有关更多信息，请参阅使用私有子网和 NAT 网关。
- 您必须对 GuardDuty 具有 `AWSServiceRoleForAmazonGuardDuty` 角色。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty 的服务相关角色权限。
- 要用运行时监控保护的任何文件都必须能够由根用户访问。如果您手动更改了文件的权限，则必须将其设置为 `755`。

以下是在 EC2 容器实例上使用运行时监控的先决条件：

- 您必须使用版本 `20230929` 或更高版本的 Amazon ECS-AMI。
- 您必须在容器实例上运行版本 `1.77` 或更高版本的 Amazon ECS 代理。
- 您必须使用内核版本 `5.10` 或更高版本。
- 有关支持的 Linux 操作系统和架构的信息，请参阅 GuardDuty 运行时监控支持哪些运营模式和工作负载。
- 您可以使用 Systems Manager 管理您的容器实例。有关更多信息，请参阅 AWS Systems Manager Session Manager 用户指南中的为 EC2 实例设置 Systems Manager。

您可以在 GuardDuty 中启用运行时监控。有关如何启用该功能的信息，请参阅《Amazon GuardDuty User Guide》中的 Enabling Runtime Monitoring。

为现有 Amazon ECS Fargate 任务添加运行时监控

开启运行时监控后，集群中所有新的独立任务和新的服务部署都自动受到保护。为了保留不可变性约束，现有的任务不受影响。

以下是使用运行时监控的先决条件：

- Fargate 平台版本必须为适用于 Linux 的 `1.4.0` 或更高版本。
- Amazon ECS 的 IAM 角色和权限：
  - Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理 GuardDuty 安全代理的权限。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色。
  - 您可以使用预定义的标签来控制集群的运行时监控。如果您的访问策略基于标签限制访问，则必须向您的 IAM 用户授予标记集群的明确权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。
- 连接到 Amazon ECR 存储库：

  GuardDuty 安全代理存储于 Amazon ECR 存储库中。每个独立任务和服务任务都必须具有访问该存储库的权限。可以使用以下选项之一：
  - 对于公有子网中的任务，您可以对任务使用公有 IP 地址，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。
  - 对于私有子网中的任务，您可以使用网络地址转换（NAT）网关，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。

    有关更多信息，请参阅使用私有子网和 NAT 网关。
- 您必须对 GuardDuty 具有 `AWSServiceRoleForAmazonGuardDuty` 角色。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty 的服务相关角色权限。
- 要用运行时监控保护的任何文件都必须能够由根用户访问。如果您手动更改了文件的权限，则必须将其设置为 `755`。

以下是在 EC2 容器实例上使用运行时监控的先决条件：

- 您必须使用版本 `20230929` 或更高版本的 Amazon ECS-AMI。
- 您必须在容器实例上运行版本 `1.77` 或更高版本的 Amazon ECS 代理。
- 您必须使用内核版本 `5.10` 或更高版本。
- 有关支持的 Linux 操作系统和架构的信息，请参阅 GuardDuty 运行时监控支持哪些运营模式和工作负载。
- 您可以使用 Systems Manager 管理您的容器实例。有关更多信息，请参阅 AWS Systems Manager Session Manager 用户指南中的为 EC2 实例设置 Systems Manager。

要立即保护任务，您需要执行以下操作之一：

- 对于独立任务，请停止任务，然后再启动它们。有关更多信息，请参阅停止 Amazon ECS 任务和将应用程序作为 Amazon ECS 任务运行
- 对于属于服务的任务，使用“强制执行新部署”选项更新服务。有关更多信息，请参阅 使用控制台更新 Amazon ECS 服务。

从 Amazon ECS 集群中移除运行时监控

您可能需要将某些集群排除在保护范围之外，例如用于测试的集群。这将导致 GuardDuty 对集群中的资源执行以下操作：

- 不再将 GuardDuty 安全代理部署到新的独立 Fargate 任务或新的服务部署中。

  为了保留不可变性约束，启用了运行时监控的现有任务和部署不受影响。
- 停止计费，且不再接受任务的运行时事件。

执行以下步骤从集群中移除运行时监控。

1. 使用 Amazon ECS 控制台或 AWS CLI 将集群上的 `GuardDutyManaged` 标签密钥设置为 `false`。有关更多信息，请参阅更新集群或使用 CLI 或 API 处理标签。为标签使用以下值。

   > **Note**
   >
   > 键和值区分大小写，并且必须与字符串完全匹配。

   键 = `GuardDutyManaged`，值 = `false`
2. 删除集群的 GuardDuty VPC 端点。有关如何删除 VPC 端点的更多信息，请参阅《AWS PrivateLink 用户指南》中的删除接口端点。

从账户中移除 Amazon ECS 的运行时监控

当您不再想要使用“运行时监控”时，在 GuardDuty 中禁用此功能。有关如何禁用此功能的信息，请参阅《Amazon GuardDuty 用户指南》中的启用运行时监控。

GuardDuty 执行以下操作：

- 删除托管集群的每个 VPC 的 GuardDuty 的 VPC 端点。
- 不再将 GuardDuty 安全代理部署到新的独立 Fargate 任务或新的服务部署中。

  为了保留不可变性约束，在停止、复制或扩展现有任务和部署前，它们不会受到影响。
- 停止计费，且不再接受任务的运行时事件。

Amazon ECS 上的 EC2 工作负载的运行时监控

当您将 EC2 实例用作容量，或者需要精细控制 Fargate 上的集群级别的运行时监控时，请使用此选项。

您可以通过添加预定义标签来预调配群集以进行运行时监控。

对于 EC2 容器实例，您可以下载、安装和管理 GuardDuty 安全代理。

对于 Fargate，GuardDuty 代表您管理安全代理。

为 Amazon ECS 开启运行时监控

您可以为带有 EC2 实例的集群开启运行时监控，或者当您需要在 Fargate 上精细控制集群级别的运行时监控时。

以下是使用运行时监控的先决条件：

- Fargate 平台版本必须为适用于 Linux 的 `1.4.0` 或更高版本。
- Amazon ECS 的 IAM 角色和权限：
  - Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理 GuardDuty 安全代理的权限。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色。
  - 您可以使用预定义的标签来控制集群的运行时监控。如果您的访问策略基于标签限制访问，则必须向您的 IAM 用户授予标记集群的明确权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。
- 连接到 Amazon ECR 存储库：

  GuardDuty 安全代理存储于 Amazon ECR 存储库中。每个独立任务和服务任务都必须具有访问该存储库的权限。可以使用以下选项之一：
  - 对于公有子网中的任务，您可以对任务使用公有 IP 地址，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。

- 对于私有子网中的任务，您可以使用网络地址转换（NAT）网关，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。

  有关更多信息，请参阅使用私有子网和 NAT 网关。
- 您必须对 GuardDuty 具有 `AWSServiceRoleForAmazonGuardDuty` 角色。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty 的服务相关角色权限。
- 要用运行时监控保护的任何文件都必须能够由根用户访问。如果您手动更改了文件的权限，则必须将其设置为 `755`。

以下是在 EC2 容器实例上使用运行时监控的先决条件：

- 您必须使用版本 `20230929` 或更高版本的 Amazon ECS-AMI。
- 您必须在容器实例上运行版本 `1.77` 或更高版本的 Amazon ECS 代理。
- 您必须使用内核版本 `5.10` 或更高版本。
- 有关支持的 Linux 操作系统和架构的信息，请参阅 GuardDuty 运行时监控支持哪些运营模式和工作负载。
- 您可以使用 Systems Manager 管理您的容器实例。有关更多信息，请参阅 AWS Systems Manager Session Manager 用户指南中的为 EC2 实例设置 Systems Manager。

您可以在 GuardDuty 中开启运行时监控。有关如何启用该功能的信息，请参阅《Amazon GuardDuty User Guide》中的 Enabling Runtime Monitoring。

向 Amazon ECS 集群添加运行时监控

配置集群的运行时监控，然后在您的 EC2 容器实例上安装 GuardDuty 安全代理。

以下是使用运行时监控的先决条件：

- Fargate 平台版本必须为适用于 Linux 的 `1.4.0` 或更高版本。
- Amazon ECS 的 IAM 角色和权限：
  - Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理 GuardDuty 安全代理的权限。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色。
  - 您可以使用预定义的标签来控制集群的运行时监控。如果您的访问策略基于标签限制访问，则必须向您的 IAM 用户授予标记集群的明确权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。
- 连接到 Amazon ECR 存储库：

GuardDuty 安全代理存储于 Amazon ECR 存储库中。每个独立任务和服务任务都必须具有访问该存储库的权限。可以使用以下选项之一：

- 对于公有子网中的任务，您可以对任务使用公有 IP 地址，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。
- 对于私有子网中的任务，您可以使用网络地址转换（NAT）网关，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。

  有关更多信息，请参阅使用私有子网和 NAT 网关。

- 您必须对 GuardDuty 具有 `AWSServiceRoleForAmazonGuardDuty` 角色。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty 的服务相关角色权限。
- 要用运行时监控保护的任何文件都必须能够由根用户访问。如果您手动更改了文件的权限，则必须将其设置为 `755`。

以下是在 EC2 容器实例上使用运行时监控的先决条件：

- 您必须使用版本 `20230929` 或更高版本的 Amazon ECS-AMI。
- 您必须在容器实例上运行版本 `1.77` 或更高版本的 Amazon ECS 代理。
- 您必须使用内核版本 `5.10` 或更高版本。
- 有关支持的 Linux 操作系统和架构的信息，请参阅 GuardDuty 运行时监控支持哪些运营模式和工作负载。
- 您可以使用 Systems Manager 管理您的容器实例。有关更多信息，请参阅 AWS Systems Manager Session Manager 用户指南中的为 EC2 实例设置 Systems Manager。

执行以下操作以将运行时监控添加到集群中。

1. 为每个集群 VPC 创建 GuardDuty 的 VPC 端点。有关更多信息，请参阅《GuardDuty 用户指南》中的手动创建 Amazon VPC 端点。
2. 配置 EC2 容器实例。

   a. 在集群中的 EC2 容器实例上，将 Amazon ECS 代理更新到版本 `1.77` 或更高版本。有关更多信息，请参阅更新 Amazon ECS 容器代理。

   b. 在集群中的 EC2 容器实例上，安装 GuardDuty 安全代理。有关更多信息，请参阅《GuardDuty 用户指南》中的手动管理 Amazon EC2 实例上的安全代理。

由于 GuardDuty 安全代理在 EC2 容器实例上作为一个进程运行，因此所有新任务和现有任务以及部署都将立即受到保护。

3. 使用 Amazon ECS 控制台或 AWS CLI 将集群上的 `GuardDutyManaged` 标签密钥设置为 `true`。有关更多信息，请参阅更新集群或使用 CLI 或 API 处理标签。为标签使用以下值。

> **Note**
>
> 键和值区分大小写，并且必须与字符串完全匹配。

键 = `GuardDutyManaged`，值 = `true`

为现有 Amazon ECS 任务添加运行时监控

开启运行时监控后，集群中所有新的独立任务和新的服务部署都自动受到保护。为了保留不可变性约束，现有的任务不受影响。

以下是使用运行时监控的先决条件：

- Fargate 平台版本必须为适用于 Linux 的 `1.4.0` 或更高版本。
- Amazon ECS 的 IAM 角色和权限：
  - Fargate 任务必须使用任务执行角色。此角色授予任务代表您检索、更新和管理 GuardDuty 安全代理的权限。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色。
  - 您可以使用预定义的标签来控制集群的运行时监控。如果您的访问策略基于标签限制访问，则必须向您的 IAM 用户授予标记集群的明确权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 教程：基于标签定义访问 AWS 资源的权限。
- 连接到 Amazon ECR 存储库：

  GuardDuty 安全代理存储于 Amazon ECR 存储库中。每个独立任务和服务任务都必须具有访问该存储库的权限。可以使用以下选项之一：

  - 对于公有子网中的任务，您可以对任务使用公有 IP 地址，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的Amazon ECR 接口 VPC 端点 (AWS PrivateLink)。
  - 对于私有子网中的任务，您可以使用网络地址转换（NAT）网关，也可以在任务运行的子网中为 Amazon ECR 创建 VPC 端点。

有关更多信息，请参阅使用私有子网和 NAT 网关。

- 您必须对 GuardDuty 具有 `AWSServiceRoleForAmazonGuardDuty` 角色。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty 的服务相关角色权限。
- 要用运行时监控保护的任何文件都必须能够由根用户访问。如果您手动更改了文件的权限，则必须将其设置为 `755`。

以下是在 EC2 容器实例上使用运行时监控的先决条件：

- 您必须使用版本 `20230929` 或更高版本的 Amazon ECS-AMI。
- 您必须在容器实例上运行版本 `1.77` 或更高版本的 Amazon ECS 代理。
- 您必须使用内核版本 `5.10` 或更高版本。
- 有关支持的 Linux 操作系统和架构的信息，请参阅 GuardDuty 运行时监控支持哪些运营模式和工作负载。
- 您可以使用 Systems Manager 管理您的容器实例。有关更多信息，请参阅 AWS Systems Manager Session Manager 用户指南中的为 EC2 实例设置 Systems Manager。

要立即保护任务，您需要执行以下操作之一：

- 对于独立任务，请停止任务，然后再启动它们。有关更多信息，请参阅停止 Amazon ECS 任务和将应用程序作为 Amazon ECS 任务运行
- 对于属于服务的任务，使用“强制执行新部署”选项更新服务。有关更多信息，请参阅 使用控制台更新 Amazon ECS 服务。

从 Amazon ECS 集群中移除运行时监控

您可以从集群中删除运行时监控。这样会导致 GuardDuty 停止监控集群中的所有资源。

要从集群中删除运行时监控。

1. 使用 Amazon ECS 控制台或 AWS CLI 将集群上的 `GuardDutyManaged` 标签密钥设置为 `false`。有关更多信息，请参阅更新集群或使用 CLI 或 API 处理标签。

> **Note**
>
> 键和值区分大小写，并且必须与字符串完全匹配。

键 = `GuardDutyManaged`，值 = `false`

2. 在集群中的 EC2 容器实例上卸载 GuardDuty 安全代理。

   有关更多信息，请参阅《GuardDuty 用户指南》中的手动卸载安全代理。

3. 删除每个集群 VPC 的 GuardDuty VPC 端点。有关如何删除 VPC 端点的更多信息，请参阅《AWS PrivateLink 用户指南》中的删除接口端点。

更新 Amazon ECS 容器实例上的 GuardDuty 安全代理

有关如何更新 EC2 容器实例上的 GuardDuty 安全代理的信息，请参阅《Amazon GuardDuty 用户指南》中的更新 GuardDuty 安全代理。

从账户中移除 Amazon ECS 的运行时监控

当您不再想要使用“运行时监控”时，在 GuardDuty 中禁用此功能。有关如何禁用此功能的信息，请参阅《Amazon GuardDuty 用户指南》中的启用运行时监控。

要从所有集群中删除运行时监控。有关更多信息，请参阅 从 Amazon ECS 集群中移除运行时监控。

运行时监控故障排除常见问题

您可能需要排查或验证运行时监控是否已启用并在您的任务和容器上运行。

主题

- 如何判断我的账户上是否激活了运行时监控？
- 如何判断集群上是否激活了运行时监控？
- 如何判断 GuardDuty 安全代理是否正在 Fargate 任务上运行？
- 如何判断 GuardDuty 安全代理是否在 EC2 容器实例上运行？
- 如果集群上运行的任务没有任务执行角色，会发生什么情况？
- 如何判断我是否拥有标记集群以进行运行时监控的正确权限？
- 如果没有连接 Amazon ECR 会发生什么情况？
- 启用运行时监控后，如何解决 Fargate 任务中的内存不足错误？

如何判断我的账户上是否激活了运行时监控？

在 Amazon ECS 控制台中，信息位于账户设置页面上。

您也可以使用 effective-settings 选项运行 list-account-settings。

```
aws ecs list-account-settings --effective-settings
```

输出

名称设置为 guardDutyActivate、值设置为 on 的设置表示账户已配置。您必须向 GuardDuty 管理员核实管理是自动的还是手动的。

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": "aws-managed"
    }
}
```

如何判断集群上是否激活了运行时监控？

在 Amazon ECS 控制台中，信息位于集群详细信息页面的标签选项卡上。

您也可以使用 TAGS 选项运行 describe-clusters。

以下示例显示默认集群的输出

```
aws ecs describe-clusters --cluster default --include TAGS
```

输出

键设置为 GuardDutyManaged、值设置为 true 的标签表示集群已配置运行时监控。

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-east-1:1234567890:cluster/default",
            "clusterName": "default",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 1,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
```

```
            "statistics": [],
            "tags": [
                {
                    "key": "GuardDutyManaged",
                    "value": "true"
                }
            ],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

如何判断 GuardDuty 安全代理是否正在 Fargate 任务上运行？

GuardDuty 安全代理作为 Fargate 任务的附加容器运行。

在 Amazon ECS 控制台中，附加容器显示在任务详细信息页面的容器下。

您可以运行 `describe-tasks` 并查找名称设置为 `aws-gd-agent`、lastStatus 设置为 `RUNNING` 的容器。

以下示例显示任务 `aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE` 的默认集群的输出。

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

输出

名为 `gd-agent` 的容器处于 `RUNNING` 状态。

```
"containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/4df26bb4-
f057-467b-a079-96167EXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/0b69d5c0-
d655-4695-98cd-5d2d5EXAMPLE",
        "lastStatus": "RUNNING",
        "healthStatus": "UNKNOWN",
```

```
        "memory": "string",
        "name": "aws-gd-agent"
      }
    ]
```

如何判断 GuardDuty 安全代理是否在 EC2 容器实例上运行？

运行以下命令以查看状态：

```
sudo systemctl status amazon-guardduty-agent
```

日志文件位于以下位置：

```
/var/log/amzn-guardduty-agent
```

如果集群上运行的任务没有任务执行角色，会发生什么情况？

对于 Fargate 任务，任务在没有 GuardDuty 安全代理附加容器的情况下启动。GuardDuty 控制面板将在覆盖率统计信息面板中显示任务缺少保护。

如何判断我是否拥有标记集群以进行运行时监控的正确权限？

为了标记集群，您必须同时对 `CreateCluster` 和 `UpdateCluster` 执行 `ecs:TagResource` 操作。

以下是示例策略的代码片段。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "ecs:CreateAction" : "CreateCluster",
             "ecs:CreateAction" : "UpdateCluster",
          }
       }
```

```
    }
  ]
}
```

如果没有连接 Amazon ECR 会发生什么情况？

对于 Fargate 任务，任务在没有 GuardDuty 安全代理附加容器的情况下启动。GuardDuty 控制面板将在覆盖率统计信息面板中显示任务缺少保护。

启用运行时监控后，如何解决 Fargate 任务中的内存不足错误？

GuardDuty 安全代理是轻量级进程。但是，该进程仍会根据工作负载的大小消耗资源。建议使用容器资源跟踪工具，例如 Amazon CloudWatch Container Insights，在集群中暂存 GuardDuty 部署。这些工具可帮助您发现应用程序的 GuardDuty 安全代理的消耗情况。然后，您可以按需要调整 Fargate 任务大小，以避免可能出现的内存不足情况。

使用 ECS Exec 监控 Amazon ECS 容器

借助 Amazon ECS Exec，您可以直接与容器交互，而无需首先与主机容器操作系统交互、打开入站端口或管理 SSH 密钥。您可以使用 ECS Exec 在 Amazon EC2 实例或 AWS Fargate 上运行的容器中运行命令或获取shell。这样，就可以更轻松地收集诊断信息并快速解决错误。例如，在开发环境中，您可以使用 ECS Exec 轻松地与容器中的各种流程进行交互，并对应用程序进行故障排除。而且，在生产场景中，您可以使用它来获得对容器的破坏式访问以调试问题。

您可以使用来自 Amazon ECS API、AWS Command Line Interface（AWS CLI）、AWS 开发工具包或 AWS Copilot CLI 的 ECS Exec 在正在运行的 Linux 或 Windows 容器中运行命令。有关使用 ECS Exec 的详细信息以及使用 AWS Copilot CLI 的视频演练，请参阅 Copilot GitHub 文档。

您还可以使用 ECS Exec 维护更严格的访问控制策略。通过选择性地启用此功能，您可以控制谁可以运行命令，以及他们可以在哪些任务上运行这些命令。通过每个命令及其输出的日志，您可以使用 ECS Exec 来查看已运行的任务，并可以使用 CloudTrail 来审核访问容器的人员。

注意事项

对于本主题，您应熟悉使用 ECS Exec 所涉及的以下方面：

- 当前不支持使用 AWS Management Console ECS Exec。
- 在以下基础设施上运行的任务支持 ECS Exec：
  - 任何经过 Amazon ECS 优化的 AMI（包括 Bottlerocket）上的 Amazon EC2 上的 Linux 容器

  - 外部实例上的 Linux 和 Windows 容器（Amazon ECS Anywhere）
  - AWS Fargate 上的 Linux 和 Windows 容器
  - Amazon EC2 上的 Windows 容器在以下经过 Windows Amazon ECS 优化的 AMI（含容器代理版本 `1.56` 或更高版本）上：
    - 经 Amazon ECS 优化的 Windows Server 2022 Full AMI
    - 经 Amazon ECS 优化的 Windows Server 2022 Core AMI
    - 经 Amazon ECS 优化的 Windows Server 2019 Full AMI
    - 经 Amazon ECS 优化的 Windows Server 2019 Core AMI
    - Amazon ECS 经过优化的 Windows Server 20H2 Core AMI
- ECS Exec 和 Amazon VPC
  - 如果您结合使用接口 Amazon VPC 端点与 Amazon ECS，则必须为 Systems Manager Session Manager（`ssmmessages`）创建接口 Amazon VPC 端点。有关 Systems Manager VPC 端点的更多信息，请参阅《AWS Systems Manager 用户指南》中的使用 AWS PrivateLink 为 Session Manager 设置 VPC 端点。
  - 如果您结合使用接口 Amazon VPC 端点与 Amazon ECS，并且使用 AWS KMS key 进行加密，则必须为 AWS KMS key 创建接口 Amazon VPC 端点。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的通过 VPC 端点连接到 AWS KMS key。
  - 当您具有在 Amazon EC2 实例上运行的任务时，请使用 `awsvpc` 联网模式。如果您无法访问互联网（未配置为使用 NAT 网关），必须为 Systems Manager Session Manager（`ssmmessages`）创建接口 Amazon VPC 端点。有关 `awsvpc` 网络模式考虑因素的更多信息，请参阅考虑因素。有关 Systems Manager VPC 端点的更多信息，请参阅《AWS Systems Manager 用户指南》中的使用 AWS PrivateLink 为 Session Manager 设置 VPC 端点。
- ECS Exec 和 SSM
  - 当用户使用 ECS Exec 在容器上运行命令时，这些命令将作为 `root` 用户运行。即使您为容器指定用户 ID，SSM Agency及其子进程以根用户身份运行。
  - SSM 代理要求容器文件系统能够写入，以便创建所需的目录和文件。因此，不支持使用 `readonlyRootFilesystem` 任务定义参数或任何其他方法将根文件系统设置为只读。
  - 虽然可以在 `execute-command` 操作之外启动 SSM 会话，但这会导致会话未被记录并根据会话限制计数。我们建议通过使用 IAM policy 拒绝 `ssm:start-session` 操作来限制此访问。有关更多信息，请参阅 限制对 “启动会话” 操作的访问。
- 以下功能作为附加容器运行。因此，您必须指定要在其上运行命令的容器名称。
  - 运行时监控
  - Service Connect

- 用户可以运行容器上下文中可用的所有命令。以下操作可能会导致孤立进程和僵尸进程：终止容器的主进程、终止命令代理和删除依赖关系。要清理僵尸进程，我们建议将 `initProcessEnabled` 标记添加到任务定义。
- ECS Exec 使用一些 CPU 和内存。在任务定义中指定 CPU 和内存资源分配时，您需要适应这一点。
- 您必须使用 AWS CLI 版本 `1.22.3` 或更高版本，或者 AWS CLI 版本 `2.3.6` 或更高版本。有关如何更新 AWS CLI 的信息，请参阅 AWS Command Line Interface 用户指南版本 2 中的安装或更新 AWS CLI 的最新版本。
- 每个进程 ID（PID）命名空间只能有一个 ECS 执行会话。如果您在任务中共享 PID 命名空间，则只能在一个容器中启动 ECS Exec 会话。
- ECS Exec 会话的空闲超时时间为 20 分钟。此值不能更改。
- 无法为现有任务启用 ECS Exec。只能为新任务启用此功能。
- 使用 `run-task` 时不能使用 ECS Exec 在使用托管扩缩和异步放置的集群上启动任务（启动没有实例的任务）。
- 您不能对 Microsoft Nano Server 容器运行 ECS Exec。

先决条件

开始使用 ECS Exec 前，请确保您已完成以下操作：

- 安装和配置 AWS CLI。有关更多信息，请参阅 AWS CLI。
- 安装 AWS CLI Session Manager 插件 有关更多信息，请参阅安装 AWS CLI Session Manager 插件。
- 您必须使用具有 ECS Exec 相应权限的任务角色。有关更多信息，请参阅任务 IAM 角色。
- ECS Exec 具有版本要求，具体取决于您的任务是托管在 Amazon EC2 上还是 AWS Fargate：
  - 如果您使用的是 Amazon EC2，必须使用 2021 年 1 月 20 日之后发布的经 Amazon ECS 优化的 AMI，代理版本为 1.50.2 或更高。有关更多信息，请参阅经 Amazon ECS 优化的 AMI。
  - 如果您使用的是 AWS Fargate，则必须使用平台版 `1.4.0` 或更高版本（Linux）或 `1.0.0`（Windows）。有关平台版本的更多信息，请参阅 AWS Fargate 平台版本。

架构

ECS Exec 使用 AWS Systems Manager (SSM) Session Manager 与正在运行的容器建立连接，并使用 AWS Identity and Access Management (IAM) 策略来控制对正在运行的容器中运行的命令的访问。

这是通过将必要的 SSM Agency 二进制文件绑定到容器中来实现的。Amazon ECS 或 AWS Fargate 代理负责启动在应用程序代码旁边的容器内的 SSM 核心代理。有关更多信息，请参阅 Systems Manager Systems Manager。

您可以使用 AWS CloudTrail 中的 `ExecuteCommand` 事件审核访问容器的用户并将每个命令（及其输出）记录到 Amazon S3 或 Amazon CloudWatch Logs。要使用自己的加密密钥加密本地客户端和容器之间的数据，必须提供 AWS Key Management Service (AWS KMS) 键。

使用 ECS Exec

可选任务定义更改

如果您将任务定义参数 `initProcessEnabled` 设置为 `true`，这将启动容器内的 init 进程，从而删除找到的任何僵尸 SSM 代理子进程。示例如下。

```
{
    "taskRoleArn": "ecsTaskRole",
    "networkMode": "awsvpc",
    "requiresCompatibilities": [
        "EC2",
        "FARGATE"
    ],
    "executionRoleArn": "ecsTaskExecutionRole",
    "memory": ".5 gb",
    "cpu": ".25 vcpu",
    "containerDefinitions": [
        {
            "name": "amazon-linux",
            "image": "amazonlinux:latest",
            "essential": true,
            "command": ["sleep","3600"],
            "linuxParameters": {
                "initProcessEnabled": true
            }
        }
    ],
    "family": "ecs-exec-task"
}
```

为任务和服务启用 ECS Exec

您可以在使用下列 AWS CLI 命令之一时，通过指定 `--enable-execute-command` 标记为您的服务和独立任务启用 ECS Exec 功能：create-service、update-service、start-task 或 run-task。

例如，如果您运行以下命令，则会为运行在 Fargate 上的新创建的服务启用 ECS Exec 功能。有关创建服务的更多信息，请参阅 create-service。

```
aws ecs create-service \
    --cluster cluster-name \
    --task-definition task-definition-name \
    --enable-execute-command \
    --service-name service-name \
    --launch-type FARGATE \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
 \
    --desired-count 1
```

为任务启用 ECS Exec 后，可以运行以下命令，确认任务是否可以使用。如果 `ExecuteCommandAgent` 的 `lastStatus` 属性列为 `RUNNING`，`enableExecuteCommand` 的属性设置为 `true`，那么您的任务就绪。

```
aws ecs describe-tasks \
    --cluster cluster-name \
    --tasks task-id
```

以下输出代码段是您可能看到的内容的示例。

```
{
    "tasks": [
        {
            ...
            "containers": [
                {
                    ...
                    "managedAgents": [
                        {
                            "lastStartedAt": "2021-03-01T14:49:44.574000-06:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
```

```
                        }
                    ]
                }
            ],
            ...
            "enableExecuteCommand": true,
            ...
        }
    ]
}
```

使用 ECS Exec 运行命令

确认 ExecuteCommandAgent 正在运行后，可以使用以下命令在容器上打开交互式 shell。如果任务包含多个容器，则必须使用 --container 标记。Amazon ECS 仅支持启动交互式会话，因此您必须使用 --interactive 标记。

以下命令将对 ID 为 *task-id* 的任务的名为 *container-name* 的容器运行一个交互式 /bin/sh 命令。

task-id 是任务的 Amazon 资源名称（ARN）。

```
aws ecs execute-command --cluster cluster-name \
    --task task-id \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

使用 ECS Exec 进行日志记录

在任务和服务中开启日志记录

> **Important**
>
> 有关 CloudWatch 定价的信息，请参阅 CloudWatch 定价。Amazon ECS 还提供了不产生额外成本的监控指标。有关更多信息，请参阅 使用 CloudWatch 监控 Amazon ECS 。

Amazon ECS 为使用 CloudWatch Logs 服务器执行运行的日志记录命令提供原定设置配置，方法是使用在任务定义中配置的 awslogs 日志驱动程序。如果您想要提供自定义配置，AWS CLI 支持 create-cluster 和 update-cluster 命令的 --configuration 标志。还有重要的一点是要知

道容器图像需要 `script` 和 `cat` 以便将命令日志正确上传到 Amazon S3 或 CloudWatch Logs。有关创建集群的更多信息，请参阅创建 create-cluster。

> **Note**
>
> 此配置仅处理 `execute-command` 会话的记录。它不会影响应用程序的日志记录。

下面的示例创建一个集群，然后将输出记录到名为 `cloudwatch-log-group-name` 的 CloudWatch Logs 日志组和名为 `s3-bucket-name` 的 Amazon S3 存储桶。

当您设置 `CloudWatchEncryptionEnabled` 选项设置为 `true` 时，您必须使用 AWS KMS 客户托管的密钥来加密日志组。。有关如何加密日志组的信息，请参阅 Amazon CloudWatch Logs 用户指南中的使用 AWS Key Management Service 加密 CloudWatch Logs 中的日志数据。

```
aws ecs create-cluster \
    --cluster-name cluster-name \
    --configuration executeCommandConfiguration="{ \
        kmsKeyId=string, \
        logging=OVERRIDE, \
        logConfiguration={ \
            cloudWatchLogGroupName=cloudwatch-log-group-name, \
            cloudWatchEncryptionEnabled=true, \
            s3BucketName=s3-bucket-name, \
            s3EncryptionEnabled=true, \
            s3KeyPrefix=demo \
        } \
    }"
```

`logging` 属性决定 ECS Exec 的日志记录功能的行为：

- `NONE`：日志记录处于关闭状态。
- `DEFAULT`：日志已发送到配置的 `awslogs` 驱动程序。如果未配置驱动程序，则不会保存任何日志。
- `OVERRIDE`：日志将发送到提供的 Amazon CloudWatch Logs 日志组和/或 Amazon S3 存储桶。

Amazon CloudWatch Logs 或 Amazon S3 日志记录所需的 IAM 权限

要启用日志记录，您的任务定义中引用的 Amazon ECS 任务角色需要具有其他权限。这些附加权限可以作为策略添加到任务角色。根据将您的日志定向到 Amazon CloudWatch Logs 还是 Amazon S3，权限会有所不同。

Amazon CloudWatch Logs

以下示例策略添加了所需的 Amazon CloudWatch Logs 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:CreateLogStream",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/
ecs/cloudwatch-log-group-name:*"
        }
   ]
}
```

Amazon S3

以下示例策略添加了所需的 Amazon S3 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
           "Effect": "Allow",
           "Action": [
```

```
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name/*"
        }
    ]
}
```

使用您自己的 AWS KMS key（KMS 密钥）加密所需的IAM权限

预设情况下，本地客户端和容器之间传输的数据使用 AWS 提供的 TLS 1.2加密。要使用自己的 KMS 密钥进一步加密数据，必须创建 KMS 密钥并将 `kms:Decrypt` 权限添加到任务 IAM 角色。您的容器将使用此权限解密数据。有关创建 KMS 密钥的更多信息，请参阅创建密钥。

您需要将以下内联策略添加到您的任务 IAM 角色，该角色需要 AWS KMS 权限。有关更多信息，请参阅 ECS Exec 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

对于要使用您自己 KMS 密钥加密的数据，必须向使用 `execute-command` 操作的用户或组授予 `kms:GenerateDataKey` 权限。

以下针对您的用户或组的示例策略包含使用您自己的 KMS 密钥所需的权限。您必须指定 KMS 密钥 Amazon Resource Name (ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

使用 IAM policy 限制对 ECS Exec 的访问权限

使用以下一个或多个 IAM 策略条件键限制用户对执行命令 API 操作的访问权限：

- aws:ResourceTag/*clusterTagKey*
- ecs:ResourceTag/*clusterTagKey*
- aws:ResourceTag/*taskTagKey*
- ecs:ResourceTag/*taskTagKey*
- ecs:container-name
- ecs:cluster
- ecs:task
- ecs:enable-execute-command

使用以下示例 IAM policy，用户可以在任务中运行的容器中运行命令，这些容器具有 environment 密钥和development 值，并在名为 cluster-name 的集群中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ExecuteCommand",
                "ecs:DescribeTasks"
```

```
            ],
         "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:ResourceTag/environment": "development"
                }
            }
        }
    ]
}
```

在以下 IAM policy 示例中，当容器名称为 `production-app` 时，用户不能使用 `execute-command` API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ecs:ExecuteCommand"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:container-name": "production-app"
                }
            }
        }
    ]
}
```

使用以下 IAM policy，用户只能在关闭 ECS Exec 时启动任务。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:enable-execute-command": "false"
                }
            }
        }
    ]
}
```

> **Note**
>
> 由于 execute-command API 操作仅包含请求中的任务和集群资源，仅评估群集和任务标签。

有关 IAM policy 条件密钥的更多信息，请参阅服务授权参考中的 Amazon Elastic Container Service 的操作、资源和条件键。

限制对 “启动会话” 操作的访问

在 ECS Exec 以外的容器上启动 SSM 会话是可能的，这可能会导致无法记录会话。在 ECS Exec 以外开始的会话也会计入会话配额。我们建议通过使用 IAM policy 直接拒绝 Amazon ECS 任务的 ssm:start-session 操作来限制此访问。您可以根据所使用的标签拒绝对所有 Amazon ECS 任务或特定任务的访问。

以下是一个 IAM policy 示例，该策略拒绝访问具有指定群集名称的所有区域中的任务的 ssm:start-session 操作。可以选择在 *cluster-name* 中包含通配符。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
```

```
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ]
        }
    ]
}
```

以下是一个 IAM policy 示例，该策略拒绝访问标记有标记键 `Task-Tag-Key` 和标记值 `Exec-Task` 的所有区域中的资源上的 `ssm:start-session` 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ecs:*:*:task/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
                }
            }
        }
    ]
}
```

有关您在使用 Amazon ECS Exec 时可能遇到的任何问题的帮助，请参阅排查 Exec 问题。

针对 Amazon ECS 的 AWS Compute Optimizer 建议

AWS Compute Optimizer 可针对 Amazon ECS 任务和容器大小生成建议。有关更多信息，请参阅AWS Compute Optimizer《用户指南》中的什么是AWS Compute Optimizer？。

为 AWS Fargate 上的 Amazon ECS 服务生成的任务和容器大小建议

AWS Compute Optimizer 可为 AWS Fargate 上的 Amazon ECS 服务生成建议。AWS Compute Optimizer 将提出有关任务 CPU 和任务内存大小以及容器 CPU、容器内存和容器内存预留大小的建议。这些建议显示在 Compute Optimizer 控制台的以下页面上。

- 为 Fargate 上的 Amazon ECS 服务生成的建议页面
- Fargate 上的 Amazon ECS 服务的详细信息页面

有关更多信息，请参阅《AWS Compute Optimizer 用户指南》中的查看为 Fargate 上的 Amazon ECS 服务生成的建议。

Amazon ECS 故障排除

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 进程守护程序和 Amazon ECS 控制台中的服务事件日志的诊断信息。

有关已停止任务的信息，请参阅以下章节之一。

操作	了解更多信息
解决已停止任务错误。	查看 Amazon ECS 已停止任务错误
查看已停止任务错误。	解决 Amazon ECS 已停止任务错误
检查已停止任务错误代码。	Amazon ECS 已停止任务错误消息
检查 CannotPullContainer 任务错误。	Amazon ECS 中的 CannotPullContainer 任务错误
查看任务 IAM 角色请求。	查看针对 Amazon ECS 任务的 IAM 角色请求

有关服务错误的信息，请参阅以下章节之一。

操作	了解更多信息
查看服务事件消息。	查看 Amazon ECS 服务事件消息
检查服务事件消息。	Amazon ECS 服务事件消息
检查负载均衡器问题。	对 Amazon ECS 中的服务负载均衡器进行故障排除

操作	了解更多信息	
检查服务自动扩缩问题。	对 Amazon ECS 中的服务自动扩缩进行故障排除	

有关任务定义错误的信息，请参阅以下章节之一。

操作	了解更多信息	
解决任务定义内存错误。	排查 Amazon ECS 任务定义 CPU 或内存无效错误	

有关 Amazon ECS 代理错误的信息，请参阅以下章节之一。

操作	了解更多信息	
查看 Amazon ECS 容器代理日志。	查看 Amazon ECS 容器代理日志	
了解如何收集 Amazon ECS 日志。	使用 Amazon ECS 日志收集器收集容器日志	
使用 Amazon ECS 代理检索诊断详细信息。	使用代理自检来检索 Amazon ECS 诊断详细信息	

有关 Docker 错误的信息，请参阅以下章节之一。

操作	了解更多信息	
使用 Docker 诊断。	Amazon ECS 中的 Docker 诊断	
开启 Docker 调试模式。	在 Amazon ECS 中配置 Docker 进程守护程序的详细输出	

操作	了解更多信息	
排查 Docker API 错误 500 问题。	对 Amazon ECS 中的 Docker API error (500): devmapper 进行故障排除	

有关 ECS Exec 和Amazon ECS Anywhere 错误的信息，请参阅以下章节之一。

操作	了解更多信息	
排查 ECS Exec 问题。	排查 Amazon ECS Exec 问题	
排查 Amazon ECS Anywhere 问题。	排查 Amazon ECS Anywhere 问题	

有关节流问题的信息，请参阅以下章节之一。

操作	了解更多信息	
了解 Fargate 节流限额。	AWS Fargate 限制配额	
了解 Amazon ECS 节流的最佳实践。	处理 Amazon ECS 节流问题	

有关 API 错误的信息，请参阅以下章节之一。

操作	了解更多信息	
解决 API 错误。	Amazon ECS API 失败原因	

解决 Amazon ECS 已停止任务错误

当任务启动失败时，控制台和 `describe-tasks` 输出参数（stoppedReason 和 stopedCode）中会显示一条错误消息。以下章节提供了有关如何解决已停止任务问题的更多信息。

以下页面提供了有关已停止任务的信息。

- 了解已停止任务错误消息的更改。

  Amazon ECS 已停止任务错误消息更新
- 查看已停止的任务，从而获取有关原因的信息。

  查看 Amazon ECS 已停止任务错误
- 了解已停止任务错误消息以及导致错误的可能原因。

  Amazon ECS 已停止任务错误消息
- 了解如何验证已停止任务连接并修正错误。

  验证 Amazon ECS 已停止任务连接

Amazon ECS 已停止任务错误消息更新

从 2024 年 6 月 14 日起，Amazon ECS 团队将更改已停止任务错误消息，如下表所述。`stopCode` 不会更改。如果您的应用程序依赖于确切的错误消息字符串，则必须使用新字符串更新应用程序。如需有关疑问或问题的帮助，请联系 AWS Support。

> **Note**
>
> 我们建议您不要依赖错误消息进行自动化，因为错误消息可能随时更改。

CannotPullContainerError

旧的错误消息。	新的错误消息	
CannotPullContainerError：来自进程守护程序的错误响应：*repository* 拉取访问被拒绝，该存储库不存在或者可能需要“docker 登录”：已拒绝：用户：*roleARN*	CannotPullContainerError：任务无法拉取映像。请检查该角色是否具有从注册表中拉取映像的权限。来自进程守护程序的错误响应：*repository* 拉取访问被拒绝，该存储库不存在或者可能需要“docker 登录”：已拒绝：用户：*roleARN*	

旧的错误消息。	新的错误消息	
	无权在资源：*image* 上执行：ecr:BatchGetImage 操作，因为没有基于身份的策略允许 ecr:BatchGetImage 操作。	
	CannotPullContainerError：任务无法拉取映像。请检查映像是否存在。来自进程守护程序的错误响应：*repository* 拉取访问被拒绝，该存储库不存在或者可能需要“docker 登录”：已拒绝：访问资源的请求已被拒绝。	
CannotPullContainerError：来自进程守护程序的错误响应：获取 *imageURI*：net/http：在等待连接期间请求被取消	CannotPullContainerError：任务无法拉取映像。请检查网络配置。来自进程守护程序的错误响应：获取 *imageURI*：net/http：在等待连接期间请求被取消（在等待标头时超出了 Client.Timeout）	

ResourceNotFoundException

旧的错误消息。	新的错误消息	
正在从 us-west-2 区域中的 AWS Secrets Manager 获取密钥数据：密钥 *sercretARN*：ResourceNotFoundException：Secrets Manager 找不到指定的密钥。	ResourceNotFoundException：任务未能在 AWS Secrets Manager 中检索到 ARN 为“*sercretARN*”的密钥。请检查指定区域中是否存在该密钥。ResourceNotFoundException：正在从 *region* 区域中的 AWS Secrets Manager 获取密	

旧的错误消息。	新的错误消息	
	钥数据：密钥 *sercretARN* ：ResourceNotFoundException：Secrets Manager 找不到指定的密钥。	

查看 Amazon ECS 已停止任务错误

如果您在启动任务时遇到问题，则您的任务可能会因应用程序或配置错误而停止。例如，在您运行任务时，该任务显示 PENDING 状态，然后消失。

如果您的任务由 Amazon ECS 服务创建，则 Amazon ECS 为维护该服务而采取的操作将在服务事件中发布。您可以在 AWS Management Console、AWS CLI、AWS 软件开发工具包、Amazon ECS API 或使用软件开发工具包和 API 的工具中查看事件。这些事件包括 Amazon ECS 停止和替换任务，因为任务中的容器已停止运行，或者 Elastic Load Balancing 的运行状况检查过多失败。

如果您的任务在 Amazon EC2 或外部计算机的容器实例上运行，您还可以查看容器运行时和 Amazon ECS 代理的日志。这些日志位于主机 Amazon EC2 实例或外部计算机上。有关更多信息，请参阅 查看 Amazon ECS 容器代理日志。

过程

Console

AWS Management Console

以下步骤可用于使用新 AWS Management Console 检查已停止任务的错误。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 在 Cluster : ***name***（集群名称：名称）页面上，选择 Tasks（任务）选项卡。
5. 将筛选器配置为显示已停止的任务。在筛选所需状态中，选择已停止或任意所需状态。

   已停止选项显示已停止的任务，任意所需状态显示您的所有任务。
6. 选择要检查的已停止任务。

7. 在已停止任务所在行的上次状态列中，选择已停止。

   弹出窗口显示停止的原因。

AWS CLI

1. 列出集群中停止的任务。输出包含您需要对任务进行描述的 Amazon Resource Name (ARN)。

```
aws ecs list-tasks \
     --cluster cluster_name \
     --desired-status STOPPED \
     --region region
```

2. 描述已停止任务以检索相关信息。有关更多信息，请参阅《AWS Command Line Interface API 参考》中的 describe-tasks。

```
aws ecs describe-tasks \
     --cluster cluster_name \
     --tasks arn:aws:ecs:region:account_id:task/cluster_name/task_ID \
     --region region
```

使用以下输出参数。

- `stopCode` – 停止代码指示任务停止的原因，例如 ResourceInitializationError
- `StoppedReason` – 任务停止的原因。
- `reason`（在 `containers` 结构中）– 原因提供了有关已停止容器的更多详细信息。

后续步骤

查看已停止的任务，从而获取有关原因的信息。有关更多信息，请参阅 Amazon ECS 已停止任务错误消息。

Amazon ECS 已停止任务错误消息

以下是任务意外停止时，您可能会收到的可能错误消息。

要使用 AWS Management Console 检查已停止的任务是否有错误消息，请参阅 查看 Amazon ECS 已停止任务错误。

已停止任务错误代码有一个与之关联的类别，例如 “ResourceInitializationError”。有关每个类别的更多信息，请参阅以下文档：

类别	了解更多信息
TaskFailedToStart	排查 Amazon ECS TaskFailedToStart 错误
ResourceInitializationError	排查 Amazon ECS ResourceInitializationError 错误
ResourceNotFoundException	排查 Amazon ECS ResourceNotFoundException 错误
SpotInterruptionError	排查 Amazon ECS SpotInterruption 错误
InternalError	排查 Amazon ECS InternalError 错误
OutOfMemoryError	排查 Amazon ECS OutOfMemoryError 错误
ContainerRuntimeError	排查 Amazon ECS ContainerRuntimeError 错误
ContainerRuntimeTimeoutError	排查 Amazon ECS ContainerRuntimeTimeoutError 错误
CannotStartContainerError	排查 Amazon ECS CannotStartContainerError 错误
CannotStopContainerError	排查 Amazon ECS CannotStopContainerError 错误
CannotInspectContainerError	排查 Amazon ECS CannotInspectContainerError 错误
CannotCreateVolumeError	排查 Amazon ECS CannotCreateVolumeError 错误

类别	了解更多信息	
CannotPullContainer	Amazon ECS 中的 CannotPullContainer 任务错误	

排查 Amazon ECS TaskFailedToStart 错误

以下是一些 `TaskFailedToStart` 错误消息和可以用来修复错误的操作。

尝试在子网“***subnet-id***”中创建启用了公有 IP 分配的网络接口时意外出现 EC2 错误

当 Fargate 任务使用 `awsvpc` 网络模式并在具有公有 IP 地址的子网中运行，且该子网没有足够的 IP 地址时，就会发生这种情况。

可用 IP 地址的数量可在 Amazon EC2 控制台中的子网详细信息页面上找到，也可以使用 `describe-subnets` 找到。有关更多信息，请参阅《Amazon VPC 用户指南》中的查看您的子网。

要解决此问题，您可以创建一个新的子网，在其中运行您的任务。

InternalError：***<reason>***

当请求 ENI 连接时会出现此错误。Amazon EC2 会异步处理 ENI 的配置。配置过程需要时间。如果等待时间长或出现未报告的故障，则 Amazon ECS 会超时。存在 ENI 已配置，但故障超时后报告发送至 Amazon ECS 的情况。在这种情况下，Amazon ECS 会出现报告的任务故障，并带有使用中的 ENI。

选定的任务定义与选定的计算策略不兼容

当您所选任务定义的启动类型与集群容量类型不匹配时，将发生此错误。有关更多信息，请参阅 Amazon ECS 启动类型。您需要选择与分配给集群的容量提供程序相匹配的任务定义。

排查 Amazon ECS ResourceInitializationError 错误

以下是一些 `ResourceInitialization` 错误消息和可以用来修复错误的操作。

无法提取密钥或注册表身份验证：任务无法从 Amazon ECR 提取注册表身份验证

当任务无法拉取任务定义中定义的映像时，将出现此错误。

可能导致此问题的原因如下：

错误原因...	请执行此操作...	
Amazon ECR VPC 端点与任务之间的网络连接问题。 在错误消息中看到以下任何字符串时，则说明问题属于网络问题： • dial tcp • dial udp • <ip>:<port>：i/o 超时 • net/http：TLS 握手超时 • 读取：连接超时 • 在等待标头时超出了 Client.Timeout • net/http：在等待连接期间请求被取消 • 信号：已终止 • 超出上下文截止时间	任务与 Amazon ECR VPC 端点之间的连接：验证 Amazon ECS 已停止任务连接。	
任务定义中定义的角色不具有 Amazon ECR 的权限。	将所需权限添加到任务执行角色。 该任务会使用以下角色之一： • 对于使用 Fargate 启动类型的任务，该任务将使用任务执行角色。有关信息，请参阅 Fargate 任务通过接口端点拉取 Amazon ECR 映像的权限。 • 对于使用 EC2 启动类型的任务，该任务将使用容器实例角色。有关信息，请参阅 Amazon ECR 权限。	

错误原因...	请执行此操作...	
映像 ARN 不存在	查看映像，然后验证以下方面： 有关查看映像的信息，请参阅《Amazon Elastic Container Registry 用户指南》中的 Viewing image details in Amazon ECR。 • 映像与任务位于同一区域。 将映像推送到正确的区域，然后用新映像 ARN 更新任务。 有关推送映像的信息，请参阅《Amazon ECR 用户指南》中的 Pushing an image to an Amazon ECR repository 有关更新任务定义的信息，请参阅《Amazon Elastic Container Service API 参考》中的 使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。 • 任务定义中的映像 ARN 不正确。 更新任务定义。有关更新任务定义的信息，请参阅《Amazon Elastic Container Service API 参考》中的 使用控制台更新 Amazon	

错误原因...	请执行此操作...	
	ECS 任务定义 或 RegisterTaskDefinition。	

无法提取密钥或注册表身份验证：无法从 ssm 检索密钥：任务无法从 Systems Manager 中提取密钥“*secretName*”

当任务无法使用 Systems Manager 中的凭证拉取任务定义中定义的映像时，将出现此错误。

可能导致此问题的原因如下：

错误原因...	请执行此操作...	
Systems Manager VPC 端点与任务之间的网络连接问题。 在错误消息中看到以下任何字符串时，则说明问题属于网络问题： • dial tcp • dial udp • <ip>:<port>：i/o 超时 • net/http：TLS 握手超时 • 读取：连接超时 • 在等待标头时超出了 Client.Timeout • net/http：在等待连接期间请求被取消 • 信号：已终止 • 超出上下文截止时间	验证任务与 Systems Manager 端点之间的连接：验证 Amazon ECS 已停止任务连接。	
任务定义中定义的角色不具有 Systems Manager 的权限。	将所需 Systems Manager 权限添加到任务执行角色。有关更多信息，请参阅 Secrets	

错误原因...	请执行此操作...	
	Manager 或 Systems Manager 权限。	
密钥 ARN 不存在	检查该 ARN 是否存在。有关更多信息，请参阅《AWS Systems Manager 用户指南》中的搜索 Systems Manager 参数。	

无法提取密钥或注册表身份验证：无法从 asm 检索密钥：任务无法从 Secrets Manager 中提取密钥“***secretARN***”

当 Fargate 任务无法使用 Secrets Manager 中的凭证拉取任务定义中定义的映像时，将出现此错误。

可能导致此问题的原因如下：

错误原因...	请执行此操作...	
Secrets Manager VPC 端点与任务之间的网络连接问题。 在错误消息中看到以下任何字符串时，则说明问题属于网络问题： • dial tcp • dial udp • <ip>:<port>：i/o 超时 • net/http：TLS 握手超时 • 读取：连接超时 • 在等待标头时超出了 Client.Timeout • net/http：在等待连接期间请求被取消 • 信号：已终止	验证任务与 Secrets Manager 端点之间的连接。有关更多信息，请参阅 验证 Amazon ECS 已停止任务连接。	

错误原因...	请执行此操作...
• 超出上下文截止时间	
任务定义中定义的任务执行角色不具有 Secrets Manager 的权限。	将所需 Secrets Manager 权限添加到任务执行角色。有关更多信息，请参阅 Secrets Manager 或 Systems Manager 权限。
密钥 ARN 不存在	检查 Secrets Manager 中是否存在该 ARN。有关查看映像的信息，请参阅《Secrets Manager 开发人员指南》中的 Find secrets in Secrets Manager。

无法提取密钥或注册表身份验证：任务无法从 Secrets Manager 中提取密钥“***secretARN***”

当任务无法使用 Secrets Manager 中的凭证拉取任务定义中定义的映像时，将出现此错误。

此错误表明 Systems Manager VPC 端点与任务之间的网络连接存在问题。

有关如何验证任务与端点之间的连接的信息，请参阅 验证 Amazon ECS 已停止任务连接。

无法下载环境变量文件：任务无法从 Amazon S3 下载环境变量文件

当任务无法从 Amazon S3 下载环境文件时，则会出现此错误。

错误原因...	请执行此操作...
任务与 Amazon S3 之间的网络连接问题。	任务与 Amazon S3 端点之间的连接：验证 Amazon ECS 已停止任务连接。
任务定义中定义的角色不具有 Amazon S3 的权限。	向该角色添加 Amazon S3 权限。有关更多信息，请参阅 Amazon S3 存储桶文件存储权限。

无法验证日志记录器参数：任务找不到任务定义中定义的 CloudWatch Logs 日志组 ***group-name***。任务与 CloudWatch 之间存在连接问题。

当您的任务无法找到您在任务定义中定义的 CloudWatch 日志组时，将发生此错误。

此错误指示任务定义中的 CloudWatch 组不存在。

您可以使用以下选项之一来修正此问题：

要使用此选项...	请执行此操作...
更新任务定义以在容器定义中包含日志组配置。	有关更新任务定义的信息，请参阅《Amazon Elastic Container Service API 参考》中的 使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。
在 CloudWatch 中创建日志组	a. 运行以下命令获取日志组名称。 `aws ecs describe-task-definition \ --task-definition task-def-name | jq -r.taskDefinition.containerDefinitions[].logConfiguration` b. 创建日志组。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的在 CloudWatch Logs 中创建日志组。

无法初始化日志记录驱动程序

当您的任务无法找到您在任务定义中定义的 CloudWatch 日志组时，将发生此错误。

此错误指示任务定义中的 CloudWatch 组不存在。

您可以使用以下选项之一来修正此问题：

要使用此选项...	请执行此操作...
更新任务定义以在容器定义中包含日志组配置。	有关更新任务定义的信息，请参阅《Amazon Elastic Container Service API 参考》中的 使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。
在 CloudWatch 中创建日志组	a. 运行以下命令获取日志组名称。 `aws ecs describe-task-definition \ --task-definition task-def-name | jq -r.taskDefinition.containerDefinitions[].logConfiguration` b. 创建日志组。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的在 CloudWatch Logs 中创建日志组。

无法调用 EFS utils 命令来设置 EFS 卷

以下问题可能会阻止您在任务中挂载 Amazon EFS 卷：

- Amazon EFS 文件系统配置不正确。
- 该任务没有所需的权限。
- 存在与网络和 VPC 配置有关的问题。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的为什么我无法在 AWS Fargate 任务上挂载 Amazon EFS 卷。

排查 Amazon ECS ResourceNotFoundException 错误

以下是一些 `ResourceNotFoundException` 错误消息和可以用来修复错误的操作。

任务未能在 AWS Secrets Manager 中检索到 ARN 为“***sercretARN***”的密钥。请检查指定区域中是否存在该密钥。

当任务无法从 Secrets Manager 中检索到密钥时，将出现此错误。这意味着 Secrets Manager 中不存在任务定义中指定（并包含在错误消息中）的密钥。

区域也包含在错误消息中。

正在从 ***region*** 区域中的 AWS Secrets Manager 获取密钥数据：密钥 ***sercretARN***：ResourceNotFoundException：Secrets Manager 找不到指定的密钥。

有关查找密钥的信息，请参阅《AWS Secrets Manager 用户指南》中的 Find secrets in AWS Secrets Manager。

可使用下表确定和解决错误。

问题	操作
该密钥位于与任务定义不同的区域。	a. 将与任务相同的区域创建该密钥。有关更多信息，请参阅创建一个 AWS Secrets Manager 密钥。 b. 使用新密钥更新任务定义。有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。
任务定义中的密钥 ARN 不正确。正确的密钥存在于 Secrets Manager 中。	使用正确的密钥更新任务定义。有关更多信息，请参阅《Amazon Elastic Container

问题	操作	
	Service API 参考》中的 使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。	
该密钥已不再存在。	a. 将与任务相同的区域创建该密钥。有关更多信息，请参阅创建一个 AWS Secrets Manager 密钥。 b. 使用新密钥更新任务定义。有关更多信息，请参阅《Amazon Elastic Container Service API 参考》中的 使用控制台更新 Amazon ECS 任务定义 或 RegisterTaskDefinition。	

排查 Amazon ECS SpotInterruption 错误

`SpotInterruption` 错误产生的原因因 Fargate 和 EC2 启动类型而异。

Fargate 启动类型

当没有 Fargate 竞价型容量或 Fargate 收回竞价型容量时，将出现 `SpotInterruption` 错误。

您可以让任务在多个可用区中运行，以提供更多容量。

EC2 启动类型

当没有可用的竞价型实例或 EC2 收回竞价型实例容量时，将出现此错误。

您可以让实例在多个可用区中运行，以提供更多容量。

排查 Amazon ECS InternalError 错误

适用于：Fargate 启动类型

当代理遇到意外的非运行时相关的内部错误时，将出现 `InternalError` 错误。

仅当使用平台版本 1.4 或更高版本时才会发生此错误。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的 How do I troubleshoot an Amazon ECS task that failed to start in an ECS cluster。

排查 Amazon ECS OutOfMemoryError 错误

以下是一些 OutOfMemoryError 错误消息和可以用来修正错误的操作。

容器因内存使用情况而被终止

当容器由于容器中的进程使用的内存超过任务定义中分配的内存而退出时，将发生此错误。

排查 Amazon ECS ContainerRuntimeError 错误

以下是一些 ContainerRuntimeError 错误消息和可以用来修正错误的操作。

ContainerRuntimeError

当代理收到来自 containerd 运行时特定操作的意外错误时，将发生此错误。此错误通常是由代理或 containerd 运行时的内部故障所导致。

仅当使用平台版本 1.4.0 或更高版本（Linux）或者 1.0.0 或更高版本（Windows）时才会发生此错误。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的为什么我的 Amazon ECS 任务停止了。

排查 Amazon ECS ContainerRuntimeTimeoutError 错误

以下是一些 ContainerRuntimeTimeoutError 错误消息和可以用来修正错误的操作。

无法过渡到正在运行状态；等待 1 分钟后超时或出现 Docker 超时错误

当容器无法在超时时间内转换到 RUNNING 或 STOPPED 状态时，会发生此错误。错误消息中将提供原因和超时值。

排查 Amazon ECS CannotStartContainerError 错误

以下是一些 CannotStartContainerError 错误消息和可以用来修正错误的操作。

无法获取容器状态：*<reason>*

容器无法启动时，将发生此错误。

排查 Amazon ECS CannotStopContainerError 错误

以下是一些 CannotStopContainerError 错误消息和可以用来修正错误的操作。

CannotStopContainerError

容器无法停止时，将发生此错误。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的为什么我的 Amazon ECS 任务停止了。

排查 Amazon ECS CannotInspectContainerError 错误

以下是一些 CannotInspectContainerError 错误消息和可以用来修正错误的操作。

CannotInspectContainerError

当容器代理无法通过容器运行时描述容器时，将发生此错误。

当使用平台版本 `1.3` 或更早版本时，Amazon ECS 代理会从 Docker 返回原因。

当使用平台版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0` 或更高版本（Windows）时，Fargate 代理从 `containerd` 返回原因。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的为什么我的 Amazon ECS 任务停止了。

排查 Amazon ECS CannotCreateVolumeError 错误

以下是一些 CannotCreateVolumeError 错误消息和可以用来修正错误的操作。

CannotCreateVolumeError

当代理无法创建任务定义中指定的卷挂载时，将发生此错误。

仅当使用平台版本 `1.4.0` 或更高版本（Linux）或者 `1.0.0` 或更高版本（Windows）时才会发生此错误。

有关如何调试和修复此问题的信息，请参阅 AWS re:Post 上的为什么我的 Amazon ECS 任务停止了。

Amazon ECS 中的 CannotPullContainer 任务错误

以下错误表明任务无法启动，因为 Amazon ECS 无法检索指定的容器映像。

> **Note**
>
> 1.4 Fargate 平台版本会截断长错误消息。

错误

- 任务无法拉取映像。请检查该角色是否具有从注册表中拉取映像的权限
- 任务无法拉取映像。检查网络配置
- API 错误（500）：获取 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/：net/http：在等待连接期间请求被取消
- API 错误
- 写入 /var/lib/docker/tmp/GetImageBlob111111111：设备上空间不足
- ERROR：toomanyrequests：请求太多或您已达到拉取速率限制。
- 来自进程守护程序的错误响应：获取 url：net/http：在等待连接期间请求被取消
- ref pull 已重试 1 次：无法复制：httpReaderSeeker：无法打开：意外状态代码
- 拉取访问被拒绝
- pull 命令失败：panic：运行时错误：内存地址无效或指针取消引用为零
- 拉取映像配置时出错
- 上下文已取消

任务无法拉取映像。请检查该角色是否具有从注册表中拉取映像的权限

此错误表明由于存在权限问题，任务无法拉取任务定义中指定的映像。错误消息中还包含有关导致问题的映像或角色的额外信息。

"来自进程守护程序的错误响应：*repository* 拉取访问被拒绝，该存储库不存在或者可能需要“docker 登录”：已拒绝：用户：*roleARN* 无权在资源：*image* 上执行：ecr:BatchGetImage 操作，因为没有基于身份的策略允许 ecr:BatchGetImage 操作。"

要解决此问题，请执行以下操作：

1. 检查 *irepository* 中是否存在该映像。有关查看映像的信息，请参阅《Amazon Elastic Container Registry 用户指南》中的 Viewing image details in Amazon ECR。
2. 验证 *role-arn* 是否具有拉取映像的适当权限。

有关如何查看和修改角色的信息，请参阅《AWS Identity and Access Management 使用指南》中的修改角色。

该任务会使用以下角色之一：

- 对于使用 Fargate 启动类型的任务，该任务将使用任务执行角色。有关其他 Amazon ECR 权限的信息，请参阅 Fargate 任务通过接口端点拉取 Amazon ECR 映像的权限。
- 对于使用 EC2 启动类型的任务，该任务将使用容器实例角色。有关其他 Amazon ECR 权限的信息，请参阅 Amazon ECR 权限。

任务无法拉取映像。检查网络配置

此错误表明任务无法连接到 Amazon ECR。

有关如何验证和解决问题的信息，请参阅 验证 Amazon ECS 已停止任务连接。

API 错误（500）：获取 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/：net/http：在等待连接期间请求被取消

此错误表明连接超时，原因是到互联网的路由不存在。

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 将应用程序作为 Amazon ECS 任务运行。
- 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，将请求路由到互联网。有关更多信息，请参阅 Amazon VPC 用户指南中的 NAT 网关。

API 错误

此错误表明 Amazon ECR 端点存在连接问题。

有关如何解决此问题的信息，请参阅 AWS Support 网站上的如何解决 Amazon ECS 中的 Amazon ECR 错误“CannotPullContainerError：API 错误”。

写入 /var/lib/docker/tmp/***GetImageBlob111111111***：设备上空间不足

此错误表示磁盘空间不足。

要解决此问题，请释放磁盘空间。

如果使用的是经 Amazon ECS 优化的 AMI，则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
```

输出示例：

```
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，可以由正在运行的容器填充根卷。如果容器使用的是没有 `json-file` 限制的原定设置 `max-size` 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
```

预设情况下，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 将 Amazon ECS 日志发送到 CloudWatch 。

ERROR：toomanyrequests：请求太多或您已达到拉取速率限制。

此错误表明有 Docker Hub 速率限制。

如果您收到以下错误之一，则可能达到 Docker Hub 速率限制：

有关 Docker Hub 费率限制的更多信息，请参阅 了解 Docker Hub 速率限制。

如果您提高了 Docker Hub 速率限制，且需要对容器实例的 Docker 拉取进行身份验证，请参阅容器实例的私有注册表身份验证。

来自进程守护程序的错误响应：获取 ***url*** ：net/http：在等待连接期间请求被取消

此错误表明连接超时，原因是到互联网的路由不存在。

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 将应用程序作为 Amazon ECS 任务运行。
- 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，将请求路由到互联网。有关更多信息，请参阅 Amazon VPC 用户指南中的 NAT 网关。

ref pull 已重试 1 次：无法复制：httpReaderSeeker：无法打开：意外状态代码

此错误表明复制映像时发生故障。

要解决此问题，请查看以下文章之一：

- 有关 Fargate 任务，请参阅 如何解决 Fargate 上 Amazon ECS 任务的 "cannotpullcontainererror" 错误。
- 有关其他任务，请参阅如何解决 Amazon ECS 任务的 "cannotpullcontainererror" 错误。

拉取访问被拒绝

此错误表明无法访问映像。

要解决此问题，您可能需要使用 Amazon ECR 对 Docker 客户端进行身份验证。有关更多信息，请参阅《Amazon ECR 用户指南》中的私有注册表身份验证。

pull 命令失败：panic：运行时错误：内存地址无效或指针取消引用为零

此错误表明由于内存地址无效或空指针取消引用而无法访问映像。

要解决此问题，请执行以下操作：

- 检查您是否有访问 Amazon S3 的安全组规则。
- 使用网关端点时，必须在路由表中添加路由才能访问端点。

拉取映像配置时出错

此错误表明已达到速率限制或存在网络错误：

要解决此问题，请参阅如何解决我的 Amazon ECS EC2 启动类型任务中的“CannotPullContainerError”错误。

上下文已取消

此错误表明上下文已取消。

此错误的常见原因是，您的任务使用的 VPC 没有从 Amazon ECR 中提取容器镜像的路径。

验证 Amazon ECS 已停止任务连接

有时，任务会因为网络连接问题而停止。这可能是间歇性问题，但很可能是因任务无法连接到端点所致。

测试任务连接

您可以使用 `AWSSupport-TroubleshootECSTaskFailedToStart` 运行手册来测试任务连接。使用运行手册时，您需要提供以下资源信息：

- 任务 ID

  使用最近失败的任务的 ID。
- 任务所在的集群

有关如何使用运行手册的信息，请参阅《AWS Systems Manager 自动化运行手册参考》中的 `AWSSupport-TroubleshootECSTaskFailedToStart`。

运行手册会对任务进行分析。您可以在输出部分查看以下可能导致任务无法启动的问题的结果：

- 与已配置的容器注册表的网络连接

- VPC 端点连接
- 安全组规则配置

修正 VPC 端点问题

当 `AWSSupport-TroubleshootECSTaskFailedToStart` 运行手册结果表明存在 VPC 端点问题时，请检查以下配置：

- 您创建该端点的 VPC 需要使用私有 DNS。
- 确认任务无法连接到的服务所用 AWS PrivateLink 端点与任务位于同一 VPC 中。有关更多信息，请参阅以下章节之一：

服务	服务的 VPC 端点信息
Amazon ECR	Amazon ECS 接口 VPC 端点（AWS PrivateLink）
Systems Manager	创建 VPC 端点
Secrets Manager	使用 AWS Systems Manager VPC 端点
CloudWatch	CloudWatch VPC 端点
Amazon S3	适用于 Amazon S3 的 AWS PrivateLink

- 为任务子网配置一条允许通过端口 443 的 HTTPS DNS（UDP 和 TCP）流量的出站规则。有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的向安全组添加规则。
- 如果子网具有网络 ACL，则需要使用以下 ACL 规则：
  - 一条允许通过端口 1024-65535 的流量的出站规则。
  - 一条允许通过端口 443 的 TCP 流量的入站规则。

  有关如何配置规则的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用网络 ACL 控制指向子网的流量。

修正网络问题

当 AWSSupport-TroubleshootECSTaskFailedToStart 运行手册结果表明存在网络问题时，请检查以下配置：

在公有子网中使用 awsvpc 网络模式的任务

根据运行手册执行以下配置：

- 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 将应用程序作为 Amazon ECS 任务运行。
- 您需要一个网关来处理互联网流量。任务子网的路由表需要有一个将流量指向该网关的路由。

  有关更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的在路由表中添加和删除路由。

网关类型	路由表目的地	路由表目标
NAT	0.0.0.0/0	NAT 网关 ID
互联网网关	0.0.0.0/0	互联网网关 ID

- 如果任务子网具有网络 ACL，则需要使用以下 ACL 规则：
  - 一条允许通过端口 1024-65535 的流量的出站规则。
  - 一条允许通过端口 443 的 TCP 流量的入站规则。

  有关如何配置规则的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用网络 ACL 控制指向子网的流量。

在私有子网中使用 awsvpc 网络模式的任务

根据运行手册执行以下配置：

- 启动任务时，对于自动分配公有 IP，选择已禁用。
- 在 VPC 中配置一个 NAT 网关，以用于将请求路由到互联网。有关更多信息，请参阅 Amazon VPC 用户指南中的 NAT 网关。
- 任务子网的路由表需要有一个将流量指向 NAT 网关的路由。

有关更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的在路由表中添加和删除路由。

网关类型	路由表目的地	路由表目标
NAT	0.0.0.0/0	NAT 网关 ID

- 如果任务子网具有网络 ACL，则需要使用以下 ACL 规则：
  - 一条允许通过端口 1024-65535 的流量的出站规则。
  - 一条允许通过端口 443 的 TCP 流量的入站规则。

  有关如何配置规则的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用网络 ACL 控制指向子网的流量。

不在公有子网中使用 awsvpc 网络模式的任务

根据运行手册执行以下配置：

- 创建集群时，对于 Amazon EC2 实例的联网下的自动分配 IP，选择开启。

  此选项将为实例的主网络接口分配一个公有 IP 地址。
- 您需要一个网关来处理互联网流量。实例子网的路由表需要有一个将流量指向该网关的路由。

  有关更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的在路由表中添加和删除路由。

网关类型	路由表目的地	路由表目标
NAT	0.0.0.0/0	NAT 网关 ID
互联网网关	0.0.0.0/0	互联网网关 ID

- 如果实例子网具有网络 ACL，则需要使用以下 ACL 规则：
  - 一条允许通过端口 1024-65535 的流量的出站规则。
  - 一条允许通过端口 443 的 TCP 流量的入站规则。

  有关如何配置规则的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用网络 ACL 控制指向子网的流量。

在私有子网中使用 awsvpc 网络模式的任务

根据运行手册执行以下配置：

- 创建集群时，对于 Amazon EC2 实例的联网下的自动分配 IP，选择关闭。
- 在 VPC 中配置一个 NAT 网关，以用于将请求路由到互联网。有关更多信息，请参阅 Amazon VPC 用户指南中的 NAT 网关。
- 实例子网的路由表需要有一个将流量指向 NAT 网关的路由。

  有关更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的在路由表中添加和删除路由。

网关类型	路由表目的地	路由表目标
NAT	0.0.0.0/0	NAT 网关 ID

- 如果任务子网具有网络 ACL，则需要使用以下 ACL 规则：
  - 一条允许通过端口 1024-65535 的流量的出站规则。
  - 一条允许通过端口 443 的 TCP 流量的入站规则。

  有关如何配置规则的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用网络 ACL 控制指向子网的流量。

查看针对 Amazon ECS 任务的 IAM 角色请求

当您在 IAM 角色中使用提供程序作为任务凭证时，提供程序的请求会保存在审计日志中。审计日志继承与容器代理日志相同的日志轮换设置。可以设置 ECS_LOG_ROLLOVER_TYPE、ECS_LOG_MAX_FILE_SIZE_MB 和 ECS_LOG_MAX_ROLL_COUNT 容器代理配置变量来影响审计日志的行为。有关更多信息，请参阅 Amazon ECS 容器代理日志配置参数。

对于容器代理版本 1.36.0 及更高版本，审计日志位于 `/var/log/ecs/audit.log`。在轮换日志时，将在日志文件名的末尾添加 *YYYY*-*MM*-*DD*-*HH* 格式的时间戳。

对于容器代理版本 1.35.0 及以前的版本，审计日志位于 `/var/log/ecs/audit.log.`*YYYY*-*MM*-*DD*-*HH*。

日志条目格式如下：

- Timestamp
- 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

您可以使用以下命令查看日志文件。

```
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
```

查看 Amazon ECS 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。您可以使用 `DescribeServices` API、AWS CLI 或 AWS Management Console 查看服务事件。

使用 Amazon ECS API 查看服务事件消息时，仅返回服务调度器中的事件。这些包括最近的任务放置和实例运行状况事件。但是，Amazon ECS 控制台显示来自以下来源的服务事件。

- 来自 Amazon ECS 服务调度器的任务放置和实例运行状况事件。这些事件的前缀为服务*#service-name#*。为了确保此事件视图有帮助，我们只显示 100 最近的事件，并忽略重复的事件消息，直到问题得到解决或过去六个小时。如果在六小时内未解决问题，您会收到另一条关于该原因的维修事件消息。
- 服务 Auto Scaling 事件。这些事件的前缀为消息。10 将显示最近扩展的事件。只有在使用 Application Auto Scaling 扩展策略配置服务时，才会发生这些事件。

可以使用以下步骤查看您当前的服务事件消息。

Console

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择集群。
3. 在 Clusters（集群）页面上，选择集群。
4. 选择要检查的服务。
5. 选择 Deployments and events（部署和事件），在 Events（事件）下查看消息。

AWS CLI

使用 describe-services 命令查看指定服务的服务事件消息。

以下 AWS CLI 示例描述了*####*群集中 *service-name* 服务，它将提供最新的服务事件消息。

```
aws ecs describe-services \
    --cluster default \
    --services service-name \
    --region us-west-2
```

Amazon ECS 服务事件消息

以下是您可能会在 Amazon ECS 控制台中看到的服务事件消息的示例：

服务（*service-name*）已达到稳定状态。

当服务正常运行且达到所需的任务数量，从而达到稳定状态时，服务计划程序将发送 `service (service-name) has reached a steady state.` 服务事件。

服务调度器会定期报告状态，因此您可能会多次收到此消息。

服务（*service-name*）无法下达任务，因为没有满足所有条件的容器实例。

当服务计划程序找不到添加其他任务的可用资源时，其会发送此事件消息。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，您会收到此错误。您应向集群添加容器实例。有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

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。

注册的端口过多

用于任务放置的最密切匹配的容器实例不能超过允许的最大保留端口限制，即每个容器实例 100 个主机端口。使用动态主机端口映射可能会修复此问题。

端口已在使用

此任务的任务定义在其端口映射中使用与已在所选容器实例上运行的任务相同的端口。服务事件消息将包含选定的容器实例 ID 作为下面消息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

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，请参阅预留 Amazon ECS Linux 容器实例内存 。

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)，且会附加到托管任务的容器实例。Amazon EC2 实例对可以附加到它们的 ENI 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 限制仍为默认值。有关更多信息，请参阅 Amazon EC2 用户指南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 增加的 Amazon ECS 容器网络接口支持的实例。

有关选择使用 awsvpcTrunking 账户设置的更多信息，请参阅 增加 Amazon ECS Linux 容器实例网络接口。

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，则无法放置该任务。

造成此错误的常见原因是您的服务使用的任务使用 awsvpc 网络模式和 EC2 启动类型。您指定的集群中没有服务创建时在 awsvpcConfiguration 中指定的同一子网中注册的容器实例。

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Amazon ECS 任务定义参数和 Amazon ECS 容器代理配置。

服务（***service-name***）无法下达任务，因为没有满足所有条件的容器实例。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 单元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。

服务（*service-name*）无法下达任务，因为没有满足所有条件的容器实例。最匹配的容器实例 *container-instance-id* 遇到错误“AGENT”（代理）。

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 容器代理日志配置参数。您还应验证代理是否正在该实例上运行。如果您使用的是经 Amazon ECS 优化的 AMI，则可以尝试通过以下命令停止并重新启动代理。

- 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI 和经 Amazon ECS 优化的 Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- 对于经 Amazon ECS 优化的 Amazon Linux AMI

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

服务（*service-name*）（实例 *instance-id*）在（elb *elb-name*）中处于运行不良状况（原因是实例连续失败的次数至少为运行状况检查的 UnhealthyThreshold 次数。）

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 对 Amazon ECS 中的服务负载均衡器进行故障排除。

服务（*service-name*）无法成功一致地启动任务。

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 Amazon ECS 服务节流逻辑。

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。

服务 (*service-name*) 操作正在受限制。稍后将重试。

由于 API 限制，此服务无法启动更多任务。一旦服务调度器能够启动更多任务，它将恢复。

要请求 API 速率限制配额增加，请打开 AWS Support 中心页面，必要时登录，然后选择创建案例。选择提高服务限制。填写并提交表格。

因为服务部署配置，服务 (*service-name*) 无法在部署期间停止或启动任务。更新 minimumHealthyPercent 或 maximumPercent 值，然后重试。

由于部署配置，此服务无法在服务部署期间停止或启动任务。部署配置由 `minimumHealthyPercent` 和 `maximumPercent` 值组成，这些值是在创建服务时定义的。这些值也可以在现有服务上更新。

`minimumHealthyPercent` 表示在部署期间或容器实例耗尽时应为服务运行的任务数的下限。这是服务所需任务数的百分比。此值四舍五入。例如，如果最小运行状况百分比为 `50`，并且所需的任务计数为 4，则调度器可以在启动两个新任务之前停止两个现有任务。同样，如果最小运行状况百分比为 75%，所需任务计数为 2，则由于结果值也为 2，调度器无法停止任何任务。

`maximumPercent` 表示在部署期间或容器实例耗尽时应为服务运行的任务数的上限。这是服务所需任务数的百分比。此值四舍五入。例如，如果最大百分比为 `200` 且所需的任务计数为 4，则调度器可以在停止 4 个现有任务之前启动 4 个新任务。同样，如果最大百分比为 `125`，且目标任务计数为三，则调度器无法启动任何任务，因为结果值也是三。

设置最小运行状况百分比或最大百分比时，应确保调度器在触发部署时可以停止或启动至少一个任务。

服务 (*service-name*) 无法放置任务。原因：您已达到可同时运行的任务数量的上限

您可以请求增加导致错误的资源的配额。有关更多信息，请参阅 服务限额。要请求提高限额，请参阅《服务限额用户指南》中的请求提高限额。

服务 (*service-name*) 无法放置任务。原因：内部错误。

以下是此错误出现的可能原因：

- 由于子网位于不受支持的可用区中，该服务无法启动任务。

  有关支持的 Fargate 区域和可用区的信息，请参阅 the section called “AWS Fargate 区域”。

  有关如何查看子网可用区的信息，请参阅 Amazon VPC 用户指南中的查看您的子网。
- 您正在尝试在 Fargate Spot 上运行使用 ARM 架构的任务定义。

服务 (*service-name*) 无法放置任务。原因：请求的 CPU 配置超出您的限制。

您可以请求增加导致错误的资源的配额。有关更多信息，请参阅 服务限额。要请求提高限额，请参阅《服务限额用户指南》中的请求提高限额。

服务 (*service-name*) 无法放置任务。原因：请求的内存配置超出您的限制。

您可以请求增加导致错误的资源的配额。有关更多信息，请参阅 服务限额。要请求提高限额，请参阅《服务限额用户指南》中的请求提高限额。

服务 (*service-name*) 无法放置任务。原因：您已达到可同时运行的 vCPU 数量的上限

AWS Fargate 正在从基于任务计数的限额过渡到基于 vCPU 的限额。

您可以请求对基于 Fargate vCPU 的限额增加限额。有关更多信息，请参阅 服务限额。要请求提高 Fargate 限额，请参阅《Service Quotas 用户指南》中的请求增加限额。

服务（*service-name*）无法达到稳定状态，因为任务集（*taskSet-ID*）无法横向缩减。原因：受保护任务的数量超过了所需的任务数量。

服务的受保护任务数量超过了所需的任务数量。您可以执行以下操作之一：

- 等到当前任务的保护到期，使它们能够被终止。
- 确定哪些任务可以停止，然后使用 `UpdateTaskProtection` API 和 `protectionEnabled` 选项将 `false` 设置为取消对这些任务的保护。
- 增加服务的所需任务计数，以大于受保护任务的数量。

服务（*service-name*）无法达到稳定状态。原因：在您的容量提供程序中未找到容器实例。

当服务计划程序找不到添加其他任务的可用资源时，其会发送此事件消息。导致出现此情况的可能原因是：

没有与集群关联的容量提供程序

使用 `describe-services` 验证您是否有与集群关联的容量提供程序。您可以更新服务的容量提供程序策略。

验证容量提供程序中是否有可用容量。如果是 EC2 启动类型，则请确保容器实例满足任务定义要求。

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，您会收到此错误。您应向集群添加容器实例。有关更多信息，请参阅 启动 Amazon ECS Linux 容器实例。

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。一次只有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。

注册的端口过多

用于任务放置的最密切匹配的容器实例不能超过允许的最大保留端口限制，即每个容器实例 100 个主机端口。使用动态主机端口映射可能会修复此问题。

端口已在使用

此任务的任务定义在其端口映射中使用与已在所选容器实例上运行的任务相同的端口。服务事件消息将包含选定的容器实例 ID 作为下面消息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

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，请参阅预留 Amazon ECS Linux 容器实例内存 。

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)，且会附加到托管任务的容器实例。Amazon EC2 实例对可以附加到它们的 ENI 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 限制仍为默认值。有关更多信息，请参阅 Amazon EC2 用户指南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 增加的 Amazon ECS 容器网络接口支持的实例。

有关选择使用 awsvpcTrunking 账户设置的更多信息，请参阅 增加 Amazon ECS Linux 容器实例网络接口。

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，则无法放置该任务。

造成此错误的常见原因是，您的服务正在使用的任务使用的是 awsvpc 网络模式和 EC2 启动类型，并且您指定的集群中没有在服务创建时在 awsvpcConfiguration 中指定的同一子网中注册的容器实例。

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Amazon ECS 任务定义参数和 Amazon ECS 容器代理配置。

服务 (*service-name*) 无法放置任务。原因：目前容量不可用。请稍后重试或在其他可用区中重试。

目前没有可用容量来运行您的服务。

您可以执行以下操作之一：

- 等待 Fargate 容量或 EC2 容器实例变为可用。
- 重新启动服务并指定其他子网。

服务（*service-name*）部署失败：任务无法启动。

您的服务中的任务无法启动。

有关如何调试已停止任务的信息，请参阅Amazon ECS 已停止任务错误消息。

服务（*service-name*）等待 Amazon ECS 代理启动时超时。请查看 /var/log/ecs/ecs-agent.log 中的日志。

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 容器代理日志配置参数。您还应验证代理是否正在该实例上运行。如果您使用的是经 Amazon ECS 优化的 AMI，则可以尝试通过以下命令停止并重新启动代理。

- 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- 对于经 Amazon ECS 优化的 Amazon Linux AMI

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

###service-name#####taskSet-ID######targetGroup-ARN）中运行状况不佳，原因是 **TARGET GROUP IS NOT FOUND**。

由于未找到目标组，该服务的任务集无法通过运行状况检查。您应删除，然后重新创建该服务。除非相应的 Amazon ECS 服务已删除，否则不要删除任何 Elastic Load Balancing 目标组。

###service-name#####taskSet-ID######targetGroup-ARN）中运行状况不佳，原因是 **TARGET IS NOT FOUND**。

由于未找到目标，该服务的任务集无法通过运行状况检查。

对 Amazon ECS 中的服务负载均衡器进行故障排除

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。

Amazon ECS 服务相关角色不存在

Amazon ECS 服务链接角色允许 Amazon ECS 服务使用 Elastic Load Balancing 器注册容器实例。将在您的账户中创建该服务相关角色。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。

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。这会导致任务被终止。

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该容器实例。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注销容器实例。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

这是负载均衡器运行状况检查的一部分。它是应用程序上的一个端点，可在应用程序运行正常时返回成功状态代码（例如 200）。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

如果您的服务使用负载均衡器，则您可以使用 AWS CLI 或开发工具包来修改负载均衡器配置。有关如何修改配置的信息，请参阅 Amazon Elastic Container Service API 参考中的 UpdateService。如果您更新服务的任务定义，则在负载均衡器中指定的容器名称和容器端口必须保留在任务定义中。

您已达到可同时运行的任务数量的上限。

对于新账户，您的配额可能低于服务配额。可以在 Service Quotas 控制台中查看您的账户服务配额。要请求提高限额，请参阅《服务限额用户指南》中的请求提高限额。

对 Amazon ECS 中的服务自动扩缩进行故障排除

Application Auto Scaling 在 Amazon ECS 部署过程中关闭横向缩减进程，并在部署完成后恢复。但是，在部署过程中，除非暂停，否则将继续发生扩展进程。有关更多信息，请参阅暂停和恢复 Application Auto Scaling 的扩缩。

排查 Amazon ECS 任务定义 CPU 或内存无效错误

使用 Amazon ECS API 或 AWS CLI 注册任务定义时，或者如果指定了无效的 `cpu` 或 `memory`，则返回以下错误。

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task.
```

> **Note**
>
> 使用 Terraform 时，可能会返回以下错误。
>
> ```
> Error: ClientException: No Fargate configuration exists for given values.
> ```

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。在任务定义中，`cpu` 值可以用 CPU 单元数或 vCPU 数表示。注册任务定义时，其将转换为指示 CPU 单元的整数。在任务定义中，`memory` 值可以用 MiB 或 GB 表示。注册任务定义时，其将转换为指示 MiB 的整数。

对于仅为 `requiresCompatibilities` 参数指定 EC2 的任务定义，支持的 CPU 值介于 256， CPU 单元（0.25 vCPU）和 16384 CPU 单元（16 vCPU）之间。内存值必须为整数，且限制取决于您使用的底层 Amazon EC2 实例上的可用内存量。

对于为 `requiresCompatibilities` 参数指定的 FARGATE 任务定义（即使也指定了 EC2），必须使用下表中的一个值。这些值决定了 CPU 和内存参数支持的值范围。

对于 Fargate 上托管的任务，下表显示了有效的 CPU 和内存组合。JSON 文件中的内存值以 MiB 为单位指定。您可以通过将 GB 值乘以 1024 来其转换为 MiB。例如 1 GB = 1024 MiB。

CPU 值	内存值	AWS Fargate 支持的操作系统
256 (.25 vCPU)	512MiB、1GB、2GB	Linux
512 (.5 vCPU)	1GB、2GB、3GB、4GB	Linux
1024 (1 vCPU)	2GB、3GB、4GB、5GB、6GB、7GB、8GB	Linux、Windows
2048 (2 vCPU)	4GB 到 16GB 之间 (以 1GB 为增量)	Linux、Windows
4096 (4 vCPU)	8GB 到 30GB 之间 (以 1GB 为增量)	Linux、Windows
8192 (8 vCPU) **Note** 此选项需要 Linux 平台 `1.4.0` 或更高版本。	16 GB 到 60 GB 之间（以 4 GB 为增量）	Linux
16384 (16vCPU) **Note** 此选项需要 Linux 平台 `1.4.0` 或更高版本。	32 GB 到 120 GB 之间（以 8 GB 为增量）	Linux

对于 Amazon EC2 上托管的任务，受支持的任务 CPU 值介于 0.25 个 vCPU 与 192 个 vCPU 之间。

> **Note**
>
> Windows 容器将忽略任务级 CPU 和内存参数。

查看 Amazon ECS 容器代理日志

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` 服务提供了日志。您可以通过 SSH 连接到容器实例来查看这些日志文件。

> **Note**
>
> 如果您不确定如何收集容器实例上的所有日志，则可使用 Amazon ECS 日志收集器。有关更多信息，请参阅 使用 Amazon ECS 日志收集器收集容器日志 。

Linux 操作系统

`ecs-init` 过程将日志存储在 `/var/log/ecs/ecs-init.log` 中。

`ecs-init.log` 文件包含有关容器代理生命周期管理、配置和引导的信息。

您可以使用以下命令查看日志文件。

```
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
```

Windows 操作系统

您可以使用适用于 Windows 的 Amazon ECS 日志收集器。有关更多信息，请参阅 Github 上的 Amazon ECS Logs Collector For Windows。

1. 连接到您的实例。
2. 打开 PowerShell，然后以管理员权限运行以下命令。这些命令会下载脚本并收集日志。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
logs-collector.ps1
.\ecs-logs-collector.ps1
```

您可以为 Amazon ECS 代理和 Docker 进程守护程序开启调试日志记录。使用此选项，脚本将可以在开启调试模式之前收集日志。该脚本将重新启动 Docker 进程守护程序和 Amazon ECS 代理，然后终止该实例上运行的所有容器。请首先清空容器实例并将所有重要任务移至其他容器实例，然后再运行以下命令。

运行以下命令以开启日志记录。

```
.\ecs-logs-collector.ps1 -RunMode debug
```

使用 Amazon ECS 日志收集器收集容器日志

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 Support 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（例如经 Amazon ECS 优化的 AMI）上为 Docker 进程守护程序和 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
- Ubuntu 16.04
- Ubuntu 18.04
- Windows Server 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。

要下载并运行适用于 Linux 的 Amazon ECS 日志收集器

1. 连接到您的容器实例。
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 进程守护程序和 Amazon ECS 容器代理启用调试模式，可将 `--mode=enable-debug` 选项添加到以下命令。这可能会重新启动 Docker 进程守护程序，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 耗尽 Amazon ECS 容器实例。

```
[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 Support 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。有关详细信息，请参阅《Amazon EC2 用户指南》中的 Connecting to Your Windows Instance。
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 进程守护程序和 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 进程守护程序，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 耗尽 Amazon ECS 容器实例。

```
.\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。

使用代理自检来检索 Amazon ECS 诊断详细信息

Amazon ECS 代理自检 API 可提供有关 Amazon ECS 代理和容器实例总体状态的信息。

您可以使用代理自检 API 来获取任务中某个容器的 Docker ID。您可以使用 SSH 连接到容器实例来使用代理自检 API。

> **Important**
>
> 您的容器实例必须具有相应的 IAM 角色，以允许访问 Amazon ECS 来到达自检 API。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。

以下示例介绍两个任务，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"
        }
    ]
}
```

在上述示例中，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 容器自检。

Amazon ECS 中的 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。

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 Amazon ECS 中的 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 Amazon ECS 中的 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**
>
> 如果使用原定设置 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 将 Amazon ECS 日志发送到 CloudWatch 。

```
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 Amazon ECS 中的 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",
...
```

在 Amazon ECS 中配置 Docker 进程守护程序的详细输出

如果您遇到与 Docker 容器或映像有关的问题，可以在 Docker 进程守护程序上打开调试模式。使用调试可以从进程守护程序中提供更详细的输出，您可以使用它来检索从容器注册表（例如 Amazon ECR）中发送的错误消息。

> **Important**
>
> 此过程是经 Amazon ECS 优化的 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。

要在经 Amazon ECS 优化的 Amazon Linux AMI 上使用 Docker 进程守护程序调试模式

1. 连接到您的容器实例。
2. 使用文本编辑器（例如 vi）打开 Docker 选项文件。对于经 Amazon ECS 优化的 Amazon Linux AMI，Docker 选项文件位于 `/etc/sysconfig/docker`。
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。

对于经 Amazon ECS 优化的 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. 保存文件并退出文本编辑器。
5. 重新启动 Docker 进程守护程序。

```
sudo service docker restart
```

您可以在一个 (扩展) 代码行中执行所有这些操作：

```
Stopping docker:                                           [  OK  ]
Starting docker: .                                   [  OK  ]
```

6. 重新启动 Amazon ECS 代理。

```
sudo service ecs restart
```

现在，您的 Docker 日志应显示更详细的输出。

```
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\"}}"
```

对 Amazon ECS 中的 Docker **API error (500): devmapper** 进行故障排除

下面的 Docker 错误表示容器实例上的精简池存储已满，并且 Docker 进程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
```

预设情况下，经 Amazon ECS 优化的 Amazon Linux AMI 从版本 2015.09.d 开始，然后后续版本中启动了 8-GiB 的卷，作为文件系统的根连接 /dev/xvda 并挂载。另外还有一个在 /dev/xvdcz 挂载并供 Docker 用于镜像和元数据存储的 22 GiB 卷。如果该存储空间被填满，则 Docker 进程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 经 Amazon ECS 优化的 Linux AMI 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：

- 要查看精简轮询信息，请在容器实例上运行以下命令：

  ```
  docker info
  ```

- （Amazon ECS 容器代理 1.8.0 及更高版本）您可以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 (预设情况下，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。
- 您可以从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镜像，然后再从中启动新容器。

  要删除未运行的容器，请在容器实例上运行下面的命令：

  ```
  docker rm $(docker ps -aq)
  ```

  要删除未使用的镜像，请在容器实例上运行下面的命令：

```
docker rmi $(docker images -q)
```

- 您可以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/"
```

排查 Amazon ECS Exec 问题

以下是故障排除说明，帮助诊断使用 ECS Exec 时出现错误的原因。

使用 Exec Checker 进行验证

ECS Exec Checker 脚本提供了一种方法来确认和验证您的 Amazon ECS 集群和任务是否符合使用 ECS Exec 功能的先决条件。ECS Exec Checker 脚本通过代表您调用各种 API 来验证您的 AWS CLI 环境和集群以及任务是否为 ECS Exec 准备好。此工具需要安装最新版本的 AWS CLI，`jq` 可用。有关更多信息，请参阅 GitHub 上的 ECS Exec Checker。

调用 **`execute-command`** 时出错

如果发生 `The execute command failed` 错误，可能是下列原因。

- 任务没有所需的权限。验证用于启动任务的任务定义是否定义了任务 IAM 角色，以及该角色是否具有所需的权限。有关更多信息，请参阅 ECS Exec 权限。
- SSM 代理未安装或未运行。
- 有一个用于 Amazon ECS 的接口 Amazon VPC 端点，但没有用于 Systems Manager Session Manager 的端点。

排查 Amazon ECS Anywhere 问题

Amazon ECS Anywhere 支持向 Amazon ECS 集群注册外部实例，如本地部署服务器或虚拟机（VM）。以下是您可能遇到的常见问题以及一般故障排除建议。

主题

- 外部实例注册问题

- 外部实例网络问题
- 在外部实例上运行任务时出现问题

外部实例注册问题

向 Amazon ECS 集群注册外部实例时，必须满足以下要求：

- 必须检索 AWS Systems Manager 激活，其中包含激活 ID 和激活码。您可以用它将外部实例注册为 Systems Manager 托管实例。请求 Systems Manager 激活时，可指定注册限制和过期日期。注册限制指定可以使用激活注册的最大实例数。注册限制的默认值为 1 实例。过期日期指定激活的过期时间。原定设置值为 24 小时。如果您用于注册外部实例的 Systems Manager 激活无效，请求新实例。有关更多信息，请参阅 将外部实例注册到 Amazon ECS 集群。
- IAM 策略用于为外部实例提供与 AWS API 操作通信所需的权限。如果未正确创建此托管策略且不包含所需的权限，则外部实例注册失败。有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色。
- Amazon ECS 提供了一个安装脚本，用于在您的外部实例上安装 Docker、Amazon ECS 容器代理和 Systems Manager Agent。如果安装脚本失败，则脚本很可能无法在同一实例上再次运行，而不会发生错误。如果发生这种情况，请按照清理过程清除 AWS 资源，这样，您可以再次运行安装脚本。有关更多信息，请参阅 注销 Amazon ECS 外部实例。

  > **Note**
  >
  > 请注意，如果成功请求安装脚本并使用了 Systems Manager 激活，则第二次运行安装脚本将使用 Systems Manager 激活。这反过来可能会让您达到该激活的注册限制。如果达到限制，必须创建一个新激活。

- 在外部实例上运行 GPU 工作负载的安装脚本时，如果没有检测到或正确配置 NVIDIA 驱动程序，将发生错误。安装脚本使用 `nvidia-smi` 命令来确认 NVIDIA 驱动程序的存在。

外部实例网络问题

要传达任何更改，您的外部实例需要与 AWS 连接的网络。如果您的外部实例失去了与 AWS 的网络连接，在手动停止之前，在您的实例上运行的任务将继续运行。和 AWS 的连接恢复时，Amazon ECS 容器代理和 Systems Manager Agent 在外部实例上使用的 AWS 凭据将自动续订。有关用来沟通外部实例和 AWS 的 AWS 域的更多信息，请参阅 联网 。

在外部实例上运行任务时出现问题

如果您的任务或容器无法在外部实例上运行，最常见的原因是网络或权限相关。如果您的容器从 Amazon ECR 中提取其镜像或配置为将容器日志发送到 CloudWatch Logs，则您的任务定义必须指定一个有效的任务执行 IAM 角色。如果没有有效的任务执行 IAM 角色，您的容器将无法启动。有关网络相关问题的更多信息，请参阅 外部实例网络问题。

> **Important**
>
> Amazon ECS 提供 Amazon ECS 日志收集工具。您可以用它从外部实例收集日志进行故障排除。有关更多信息，请参阅 使用 Amazon ECS 日志收集器收集容器日志 。

AWS Fargate 限制配额

AWS Fargate 在每个区域使用每个 AWS 账户的令牌存储桶算法将 Amazon ECS 任务和 Amazon EKS pod 启动率限制为配额（以前称为限制）。使用此算法，您的账户拥有一个持有特定数量的令牌的存储桶。存储桶中的令牌数表示您在任何给定秒钟的速率配额。每个客户账户都有一个任务和 pod 令牌存储桶，该存储桶根据客户账户启动的任务和 pod 数量消耗。此令牌存储桶具有存储桶最大限额，允许您定期提出更多请求数量，以及允许您在需要的时间内维持稳定的请求速率的重填速率。

例如，Fargate 客户账户的任务和 pod 令牌存储桶大小为 100 个令牌，重填速率为每秒 20 个令牌。因此，您可以立即为每个客户账户最多启动 100 个 Amazon ECS 任务和 Amazon EKS Pod，持续启动率为每秒 20 个 Amazon ECS 任务和 Amazon EKS pod。

操作	存储桶最大容量（或突发速率）	存储桶重填速率（或持续速率）
按需 Amazon ECS 任务和 Amazon EKS pod 的 Fargate 资源率配额[1]	100	20
Spot Amazon ECS 任务的 Fargate 资源率配额	100	20

[1]使用在 Amazon EKS 平台版本中调用的平台版本时，仅启动 Amazon EKS pod 的账户的突增速率为 20，持续的 pod 启动速率为每秒 20 个 pod 启动。

在 Fargate 中对 **RunTask** API 进行节流

此外，在使用 Amazon ECS RunTask API 启动任务时，Fargate 会使用单独的配额限制请求速率。Fargate 按区域为每个 AWS 账户限制 Amazon ECS RunTask API 请求。您发出的每个请求都会从存储桶中删除一个令牌。我们这样做是为了帮助提高服务的性能，并确保所有 Fargate 客户的公平使用。API 调用受请求配额的约束，无论是来自 Amazon Elastic Container Service 控制台、命令行工具还是第三方应用程序。调用 Amazon ECS RunTask API 的速率配额为每秒 20 次调用（突发和持续）。但是，每次调用此 API 最多可以启动 10 个任务。这意味着您可以在一秒钟内启动 100 个任务，方法是对此 API 进行 10 次调用，请求在每次调用中启动 10 个任务。同样，您也可以对此 API 进行 20 次调用，请求在每次调用中启动 5 个任务。有关 Amazon ECS RunTask API 中的 API 节流的更多信息，请参阅 Amazon ECS API 参考中的 API 请求节流。

实际上，任务和 pod 启动率还取决于其他考虑因素，例如要下载和解压的容器镜像、启用的运行状况检查和其他集成，例如向负载均衡器注册任务或 pod。根据客户启用的功能，客户会看到任务和 pod 启动率与之前配额相比的变化。

在 Fargate 中调整速率配额

您可以请求增加您的 AWS 账户的 Fargate 速率限制配额。要请求配额调整，请联系 AWS Support 中心。

处理 Amazon ECS 节流问题

节流错误分为两大类：同步节流和异步节流。

同步节流

发生同步节流时，您会立即收到来自 Amazon ECS 的错误响应。如果在运行任务或创建服务时调用 Amazon ECS API，则通常会发生此类节流。有关所涉及的节流和相关节流限制的更多信息，请参阅请求对 Amazon ECS API 进行节流。

例如，当您的应用程序通过使用 AWS CLI 或 AWS SDK 发起 API 请求时，您可以修复 API 节流。为此，您可以设计应用程序以处理错误，也可以通过对 API 调用实现带有重试逻辑的指数回退和抖动策略。有关更多信息，请参阅超时、重试和回退并抖动。

如果您使用 AWS SDK，则自动重试逻辑已内置且可配置。

Amazon ECS 中的异步节流

发生异步节流是因为异步工作流程中 Amazon ECS 或 AWS CloudFormation 可能代表您调用 API 来预置资源。重要的是了解 Amazon ECS 代表您调用了哪些 AWS API。例如，对使用 awsvpc 网络模式的任务调用 CreateNetworkInterface API，对注册到负载均衡器的任务执行运行状况检查时调用 DescribeTargetHealth API。

当您的工作负载达到相当大的规模时，这些 API 操作可能会受到限制。也就是说，其可能会受到足够的限制，以违反所调用的 Amazon ECS 或 AWS 服务 强制执行的限制。例如，如果您部署了数百项服务，每项服务同时有数百个使用 awsvpc 网络模式的任务，则 Amazon ECS 会调用 Amazon EC2 API 操作（例如 CreateNetworkInterface）和 Elastic Load Balancing API 操作（例如 RegisterTarget 或 DescribeTargetHealth）分别注册弹性网络接口和负载均衡器。这些 API 调用可能会超出 API 限制，从而导致节流错误。以下是服务事件消息中包含的 Elastic Load Balancing 节流错误的示例。

```
{
   "userIdentity":{
      "arn":"arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForECS/ecs-service-
scheduler",
      "eventTime":"2022-03-21T08:11:24Z",
      "eventSource":"elasticloadbalancing.amazonaws.com",
      "eventName":" DescribeTargetHealth ",
      "awsRegion":"us-east-1",
      "sourceIPAddress":"ecs.amazonaws.com",
      "userAgent":"ecs.amazonaws.com",
      "errorCode":"ThrottlingException",
      "errorMessage":"Rate exceeded",
      "eventID":"0aeb38fc-229b-4912-8b0d-2e8315193e9c"
   }
}
```

当这些 API 调用与您账户中的其他 API 流量共享限制时，即使其作为服务事件发出，也可能难以监控。

监控节流

确定哪些 API 请求受到限制，以及发出这些请求的人员非常重要。您可以使用 AWS CloudTrail 监控节流，并与 CloudWatch、Amazon Athena 和 Amazon EventBridge 集成。您可以将 CloudTrail 配置为将特定事件发送到 CloudWatch Logs。CloudWatch Logs 日志见解可解析和分析事件。这可识别节

流事件中的详细信息，例如发出调用的用户或 IAM 角色以及发出的 API 调用数。有关更多信息，请参阅使用 CloudWatch Logs 监控 CloudTrail 日志文件。

有关 CloudWatch Logs Insights 的更多信息以及如何查询日志文件的说明，请参阅使用 CloudWatch Logs Insights 分析日志数据。

借助 Amazon Athena，您可以使用标准 SQL 创建查询和分析数据。例如，您可以创建一个 Athena 表来解析 CloudTrail 事件。有关更多信息，请参阅使用 CloudTrail 控制台为 CloudTrail 日志创建 Athena 表。

创建 Athena 表后，您可以使用简单 SQL 查询（例如以下查询）来调查 `ThrottlingException` 错误。

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
FROM cloudtrail-table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2022-01-14T03:00:08Z' and '2022-01-23T07:15:08Z'
group by errorcode, awsregion, eventsource, username, eventname
order by count desc;
```

Amazon ECS 还会向 Amazon EventBridge 发送事件通知。有资源状态变更事件和服务操作事件。它们包括 API 节流事件，例如 `ECS_OPERATION_THROTTLED` 和 `SERVICE_DISCOVERY_OPERATION_THROTTLED`。有关更多信息，请参阅 Amazon ECS 服务操作事件。

AWS Lambda 等服务可以使用这些事件来执行响应操作。有关更多信息，请参阅 处理 Amazon ECS 事件。

如果您运行独立任务，则某些 API 操作（例如 `RunTask`）是异步的，并且不会自动执行重试操作。在这种情况下，您可以使用 AWS Step Functions 等服务与 EventBridge 集成来重试受限制或失败的操作。有关更多信息，请参阅管理容器任务（Amazon ECS、Amazon SNS）。

使用 CloudWatch 监控节流

CloudWatch 为按 AWS 资源下的 `Usage` 命名空间提供 API 使用情况监控。这些指标记录为 API 类型和指标名称 CallCount。您可以创建警报，以在这些指标达到特定阈值时启动。有关更多信息，请参阅可视化 Service Quotas 并设置警报。

CloudWatch 还提供异常检测功能。此功能使用机器学习，根据您启用该功能的指标的特定行为来分析和建立基准。如果出现异常的 API 活动，则您可以将此功能与 CloudWatch 警报一起使用。有关更多信息，请参阅使用 CloudWatch 异常检测。

通过主动监控节流错误，您可以联系 AWS Support 提高相关的节流限制，还可以获得针对您独特应用需求的指导。

Amazon ECS API 失败原因

如果您通过 Amazon ECS API、控制台或 AWS CLI 退出时出现 `failures` 错误消息，以下内容可能有助于解决原因。失败会返回原因和与故障关联资源的 Amazon 资源名称（ARN）。

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。使用 AWS CLI 时，确保您的 AWS CLI 命令发送到带 `--region` *region*参数 的正确区域

有关 `Failure` 数据类型结构的更多信息，请参阅 Amazon Elastic Container Service API 参考中的失败。

以下是运行 API 命令时可能会收到的失败消息的示例。

API 操作	失败原因或停止原因	原因
`DescribeClusters`	`MISSING`	未找到指定的集群。验证集群名称的拼写。
`DescribeInstances`	`MISSING`	未找到指定的容器实例。验证您指定了容器实例注册到的集群，以及容器实例 ARN 或 ID 是否正确。
`DescribeServices`	`MISSING`	未找到指定的服务。验证是否指定了正确的群集或区域，并且服务 ARN 或名称是否有效。
`DescribeTasks`	`MISSING`	未找到指定的任务。验证是否指定了正确的集群或区域，以及任务 ARN 或 ID 都有效。
`DescribeTasks`	`TaskFailedToStart: RESOURCE:*`	对于 `RESOURCE:CPU` 错误，任务请求的 CPU 数量在给定容器实例上不可用。当任务定义中的 CPU 单位要求大于映射到容量提供程序的自动扩缩组

API 操作	失败原因或停止原因	原因
		中定义的 Amazon EC2 实例的 CPU 大小时，通常会发生这种情况。您需要检查容量提供程序的配置。 对于 RESOURCE:MEMORY 错误，任务请求的内存量在给定容器实例上不可用。当任务定义中的内存量要求大于映射到容量提供程序的自动扩缩组中定义的 Amazon EC2 实例支持的内存时，通常会发生这种情况。您需要检查容量提供程序的配置。
	TaskFailedToStart: AGENT	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。 有关如何排除断开连接的代理故障的信息，请参阅如何排除断开连接的 Amazon ECS 代理的故障。
	TaskFailedToStart: MemberOf placement constraint unsatisfied	没有满足任务定义中定义的放置约束的容器实例。

API 操作	失败原因或停止原因	原因
	TaskFailedToStart: ATTRIBUTE	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Amazon ECS 任务定义参数和 Amazon ECS 容器代理配置。
	TaskFailedToStart: NO ACTIVE INSTANCES	您的容量提供程序中没有活动实例。有关如何管理自动扩缩组的信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的自动扩缩组。
	TaskFailedToStart: EMPTY CAPACITY PROVIDER	您的集群中没有实例。这很可能是因为容量提供程序为空，或者因为容量提供程序中的实例未注册到集群。有关如何管理自动扩缩组的信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的自动扩缩组。
GetTaskProtection	MISSING	未找到指定的任务。验证集群名称或 ARN 以及任务 ARN 或 ID 是否有效。

API 操作	失败原因或停止原因	原因
	TASK_NOT_VALID	指定任务并非 Amazon ECS 服务的一部分。只有 Amazon ECS 服务托管的任务可以受到保护。验证任务 ARN 或 ID，然后重试。
RunTask 或 StartTask	RESOURCE:*	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。 如果出现 RESOURCE:ENI 错误，则说明您的群集没有任何可用的弹性网络接口附加点，这是使用 awsvpc 网络模式的任务必需的。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。要详细了解每种实例类型支持的网络接口数量，请参阅《Amazon EC2 用户指南》中的每种实例类型的每个网络接口的 IP 地址数。 对于 RESOURCE:GPU 错误，任务请求的 GPU 数量不可用，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 适用于 GPU 工作负载的 Amazon ECS 任务定义。

API 操作	失败原因或停止原因	原因
	AGENT	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。 有关如何排除断开连接的代理故障的信息，请参阅如何排除断开连接的 Amazon ECS 代理的故障。
	LOCATION	您尝试启动任务的容器实例位于与您在 awsVpcConfiguration 中指定的子网不同的可用区。
	ATTRIBUTE	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Amazon ECS 任务定义参数和 Amazon ECS 容器代理配置。
StartTask	MISSING	找不到您尝试启动任务的容器实例。检查是否指定了错误的集群或区域，或者容器实例 ARN 或 ID 是否拼写错误。
	INACTIVE	您尝试启动任务的容器实例之前已从 Amazon ECS 注销且无法使用。

API 操作	失败原因或停止原因	原因
UpdateTaskProtection	DEPLOYMENT_BLOCKED	无法设置任务保护，因为一项或多项受保护的任务正在阻止服务部署达到稳定状态。取消对现有任务的任务保护或等到任务保护到期。
	MISSING	未找到指定的任务。验证集群名称或 ARN 以及任务 ARN 或 ID 是否有效。
	TASK_NOT_VALID	指定任务并非 Amazon ECS 服务的一部分。只有 Amazon ECS 服务托管的任务可以受到保护。验证任务 ARN 或 ID，然后重试。

> **Note**
>
> 除了此处描述的故障场景外，API 操作也可能由于异常而失败，从而导致错误响应。有关此类异常的列表，请参阅 Common Errors（常见错误）。

Amazon Elastic Container Service 中的安全

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业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。责任共担模式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，第三方审核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。

该文档帮助您了解如何在使用 Amazon ECS 时应用责任共担模式。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。您还会了解如何使用其他 AWS 服务以帮助您监控和保护 Amazon ECS 资源。

主题

- Amazon Elastic Container Service 的身份和访问管理
- Amazon Elastic Container Service 中的日志记录和监控
- Amazon Elastic Container Service 的合规性验证
- AWS Fargate 美国联邦信息处理标准（FIPS-140）
- Amazon Elastic Container Service 的基础设施安全
- Amazon ECS 任务和容器安全性最佳实践

Amazon Elastic Container Service 的身份和访问管理

AWS Identity and Access Management（IAM）是一项 AWS 服务，可以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）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
- 使用身份进行身份验证
- 使用策略管理访问
- 如何将 Amazon Elastic Container Service 与 IAM 结合使用
- Amazon Elastic Container Service 的基于身份的策略示例
- Amazon Elastic Container Service AWS 托管策略
- 对 Amazon ECS 使用服务相关角色
- 适用于 Amazon ECS 的 IAM 角色
- Amazon ECS 控制台所需的权限
- Amazon ECS 服务自动扩缩所需的 IAM 权限
- 在创建时授予标记资源的权限
- 对 Amazon Elastic Container Service 标识和访问进行故障排除
- Amazon EC2 的 IAM 最佳实践

受众

AWS Identity and Access Management (IAM) 的使用方式因您可以在 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，您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon ECS 功能来完成工作，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon ECS 中的功能，请参阅 对 Amazon Elastic Container Service 标识和访问进行故障排除。

服务管理员 – 如果您在公司负责管理 Amazon ECS 资源，您可能对 Amazon ECS 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon ECS 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 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 如何将 Amazon Elastic Container Service 与 IAM 结合使用。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon ECS 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon ECS 基于身份的策略示例，请参阅 Amazon Elastic Container Service 的基于身份的策略示例。

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。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center（IAM Identity Center）用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的 如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能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《AWS IAM Identity Center 用户指南》中的多重身份验证和《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 和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户 根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、Web 身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户 时，他们代入角色，而角色提供临时凭证。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和组，也可以连接并同步到您自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使

用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

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。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。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，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

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 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以担任角色。有关使用角色的方法的更多信息，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：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。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。

- 跨服务访问 – 某些 AWS 服务 使用其他 AWS 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 AWS 服务 关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 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。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。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 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，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 AWS 身份或资源，以控制 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。在主体（用户、根用户或角色会话）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 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎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 Management Console、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、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色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式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到 AWS 账户 中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式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的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管式策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（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）概览。

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 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体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 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 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 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 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 如何确定在涉及多种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。

如何将 Amazon Elastic Container Service 与 IAM 结合使用

在使用 IAM 管理对 Amazon ECS 的访问权限之前，了解哪些 IAM 功能可用于 Amazon ECS。

IAM 功能可与 Amazon Elastic Container Service 一起使用

IAM 功能	Amazon ECS 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	部分
策略条件键	是

IAM 功能	Amazon ECS 支持
ACL	否
ABAC（策略中的标签）	是
临时凭证	是
转发访问会话（FAS）	是
服务角色	是
服务相关角色	是

要大致了解 Amazon ECS 和其它 AWS 服务如何与大多数 IAM 功能一起使用，请参阅 IAM 用户指南中的与 IAM 一起使用的 AWS 服务。

Amazon ECS 基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色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。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Amazon ECS 的基于身份的策略示例

要查看 Amazon ECS 基于身份的策略的示例，请参阅 Amazon Elastic Container Service 的基于身份的策略示例。

Amazon ECS 内基于资源的策略

支持基于资源的策略	否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源处于不同的AWS账户中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，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。

Amazon ECS 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 Amazon ECS 操作的列表，请参阅服务授权参考中的 Amazon Elastic Container Service 定义的操作。

Amazon ECS 中的策略操作在操作前面使用以下前缀：

```
ecs
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"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 基于身份的策略的示例，请参阅 Amazon Elastic Container Service 的基于身份的策略示例。

Amazon ECS 的策略资源

支持策略资源	部分

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 Amazon ECS 资源类型及其 ARN 的列表，请参阅服务授权参考中的 AAmazon Elastic Container Service 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon Elastic Container Service 定义的操作。

某些 Amazon ECS API 操作支持多个资源。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 ARN。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

例如，Amazon ECS 集群资源具有以下 ARN：

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

要在语句中指定 my-cluster-1 和 my-cluster-2 集群，请使用以下 ARN：

```
"Resource": [
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-1",
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-2"
```

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/*"
```

对于任务定义，您可以指定最新修订版本或特定修订版本。

要指定任务定义的所有修订版，请使用通配符（*）：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:*"
```

要指定特定的任务定义修订版，请使用 ${TaskDefinitionRevisionNumber}：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:${TaskDefinitionRevisionNumber}"
```

要查看 Amazon ECS 基于身份的策略的示例，请参阅 Amazon Elastic Container Service 的基于身份的策略示例。

Amazon ECS 的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

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。您可以创建使用条件运算符（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，只有在使用 IAM 用户名标记 IAM 用户时，您才能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。

AWS支持全局条件键和特定于服务的条件键。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 支持以下特定于服务的条件键，您可以使用这些键来为 IAM policy 提供精细筛选：

条件键	描述	评估类型
aws:RequestTag/${TagKey}	上下文键的格式为 "aws:RequestTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是标签键值对。 检查 AWS 请求中是否存在目标键值对。例如，您可以检查请求是否包含标签键 "Dept" 并具有 "Accounting" 值。	String
aws:ResourceTag/${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。	String
aws:TagKeys	该上下文密钥的格式为 "aws:TagKeys":" *tag-key*"，其中 *tag-key* 是没有值的标签键列表 (例如，["Dept","Cost-Center"])。	String

条件键	描述	评估类型
	检查在 AWS 请求中包含的标签键。	
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。	String
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:container-name	上下文密钥已格式化为 "ecs:container-name":" *container-name* "，其中 *container-instance-* 是任务定义中定义的 Amazon ECS 容器的名称。	String
ecs:enable-execute-command	上下文密钥已格式化为 "ecs:enable-execute-command":" *value*"，其中 *value-* 为“true”或“false”。	String
ecs:enable-service-connect	上下文键的格式为 "ecs:enable-service-connect":" *value*"，其中 *value* 为 "true" 或 "false"。	String
ecs:enable-ebs-volumes	上下文键的格式为 "ecs:enable-ebs-volumes":" *value*"，其中 *value* 为 "true" 或 "false"。	String
ecs:namespace	上下文键的格式为 "ecs:namespace":" *namespace-arn*"，其中 *namespace-arn* 为 AWS Cloud Map 命名空间的 ARN。	ARN，Null

条件键	描述	评估类型
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
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:account-setting	上下文密钥已格式化为 "ecs:account-setting":" *account-setting* "，其中 *account-setting* 是 Amazon ECS 账户设置的名称。	String

要查看 Amazon ECS 条件键的列表，请参阅服务授权参考中的 Amazon Elastic Container Service 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Elastic Container Service 定义的操作。

要查看 Amazon ECS 基于身份的策略的示例，请参阅 Amazon Elastic Container Service 的基于身份的策略示例。

Amazon ECS 中的访问控制列表（ACL）

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Amazon ECS 的基于属性的访问控制（ABAC）

> **Important**
>
> Amazon ECS 支持对所有 Amazon ECS 资源进行基于属性的访问控制。要确定是否可以使用属性来确定操作的范围，请使用《服务授权参考》中的由 Amazon ECS 定义的操作表。首先验证资源列中是否有资源。然后，使用条件键列查看操作/资源组合的键。

支持 ABAC（策略中的标签）	是

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在 AWS 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及AWS资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

有关标记 Amazon ECS 资源的更多信息，请参阅 为 Amazon ECS 资源添加标签。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅基于标签描述 Amazon ECS 服务。

将临时凭证用于 Amazon ECS

支持临时凭证	是

某些 AWS 服务在您使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务与临时凭证配合使用，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其他方法登录到AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问AWS时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问AWS。AWS 建议您动态生成临时凭证，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon ECS 的转发访问会话

支持转发访问会话 (FAS)	是

当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon ECS 的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务 委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon ECS 的功能。仅当 Amazon ECS 提供相关指导时才编辑服务角色。

Amazon ECS 的服务相关角色

支持服务相关角色	是

服务相关角色是一种与 AWS 服务 相关的服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 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。

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。

Amazon Elastic Container Service 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon ECS 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface（AWS CLI）或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 创建 IAM 策略。

有关 Amazon ECS 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅服务授权参考中的 Amazon Elastic Container Service 的操作、资源和条件键。

主题

- Amazon ECS 的策略最佳实践
- 允许 Amazon ECS 用户查看他们自己的权限
- Amazon ECS 集群示例
- Amazon ECS 容器实例示例
- Amazon ECS 任务定义示例
- 运行 Amazon ECS 任务示例
- 启动 Amazon ECS 任务示例
- 列出并描述 Amazon ECS 任务示例
- 创建 Amazon ECS 服务示例
- 更新 Amazon ECS 服务示例
- 基于标签描述 Amazon ECS 服务
- 拒绝 Amazon ECS Service Connect 命名空间覆盖示例

Amazon ECS 的策略最佳实践

基于身份的策略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：

- AWS 托管策略及转向最低权限许可入门 – 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的 AWS 托管策略。

- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定（AWS 服务例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证（MFA）– 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

允许 Amazon ECS 用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。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 集群示例

以下 IAM policy 授予创建和列出群集所需的权限。`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 policy 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:DescribeClusters",
                "ecs:DeleteCluster"
            ],
            "Resource": ["arn:aws:ecs:us-east-1:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM policy 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"}
            }
        }
    ]
}
```

Amazon ECS 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注销实例。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 policy 可让用户列出并注销指定群集中的容器类型：

```
{
    "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 policy 可让用户描述指定群集中的指定容器实例：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/
<cluster_name>/<container_instance_UUID>"]
        }
    ]
}
```

Amazon ECS 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：

如果您使用控制台，则必须添加 `CloudFormation: CreateStack` 作为 `Action`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RegisterTaskDefinition",
                "ecs:ListTaskDefinitions",
                "ecs:DescribeTaskDefinition"
            ],
            "Resource": ["*"]
        }
    ]
}
```

运行 Amazon ECS 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 policy 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 Amazon ECS 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 policy 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/<cluster_name>/
<container_instance_UUID>"]
```

```
                }
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:task-definition/
<task_family>:*"]
        }
    ]
}
```

列出并描述 Amazon ECS 任务示例

以下 IAM policy 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": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM policy 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/<cluster_name>/
<task_UUID>"]
        }
```

```
    ]
}
```

创建 Amazon ECS 服务示例

以下 IAM policy 可让用户在 AWS Management Console 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:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

更新 Amazon ECS 服务示例

以下 IAM policy 可让用户在 AWS Management Console 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: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}"}
            }
        }
    ]
}
```

您可以将该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，将拒绝其访问。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：条件。

拒绝 Amazon ECS Service Connect 命名空间覆盖示例

以下 IAM policy 拒绝用户覆盖服务配置中的默认 Service Connect 命名空间。默认命名空间在集群中设置。然而，您可以在服务配置中将其覆盖。为了保持一致性，可以考虑将所有新服务设置为使用相同的命名空间。使用以下上下文键要求服务使用特定的命名空间。在以下示例中，将 `<region>`、`<aws_account_id>`、`<cluster_name>` 和 `<namespace_id>` 替换为自己的内容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:namespace":
 "arn:aws:servicediscovery:<region>:<aws_account_id>:namespace/<namespace_id>"
                }
            },
            "Resource": "*"
        }
    ]
}
```

Amazon Elastic Container Service AWS 托管策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 AWS 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。

AWS 服务负责维护和更新 AWS 托管式策略。您无法更改 AWS 托管式策略中的权限。服务偶尔会向 AWS 托管式策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 AWS 托管式策略。服务不会从 AWS 托管策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，AWS 还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动新特征时，AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的 AWS 托管式策略。

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用户、组、角色、Amazon 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。有关 Amazon ECR 托管策略的更多信息，请参阅 Amazon ECR 托管策略。

AmazonECS_FullAccess

您可以将 `AmazonECS_FullAccess` 策略附加到 IAM 身份。

此策略授予对 Amazon ECS 资源的管理访问权限，并授予 IAM 身份（如用户、组或角色）访问 AWS 服务，以使用 Amazon ECS 的所有功能。使用此策略可以访问 AWS Management Console 中提供的所有 Amazon ECS 功能。

权限详细信息

`AmazonECS_FullAccess` 托管 IAM policy 必须包含以下权限。遵循授予最低权限的最佳实践，您可以使用 `AmazonECS_FullAccess` 托管策略作为创建您自己的自定义策略的模板。这样，您就可以根据您的特定要求取消或添加托管策略的权限。

- `ecs` – 允许主体完全访问所有 Amazon ECS API 操作。
- `application-autoscaling`— 允许委托人创建、描述和管理 Application Auto Scaling 资源。为您的 Amazon ECS 服务启用服务自动扩展时，此操作是必需的。

- appmesh— 允许委托人列出 App Mesh 服务网格和虚拟节点，并描述 App Mesh 虚拟节点。将您的 Amazon ECS 服务与 App Mesh 集成时需要这样做。
- autoscaling— 允许委托人创建、托管和描述 Amazon EC2 Auto Scaling 资源。使用集群自动扩缩功能时，在管理 Amazon EC2 Auto Scaling 组时需要此项。
- cloudformation— 允许委托人创建和管理 AWS CloudFormation 堆栈。这是使用 AWS Management Console 创建 Amazon ECS 群集和后续管理这些群集时所必需的。
- cloudwatch— 允许委托人创建、托管和描述 Amazon CloudWatch 警报。
- codedeploy – 允许委托人创建和管理应用程序部署并查看其配置、修订和部署目标。
- sns— 允许委托人查看 Amazon SNS 主题列表。
- lambda— 允许委托人查看 AWS Lambda 函数及其版本特定的配置。
- ec2 – 允许委托人运行 Amazon EC2 实例以及创建和管理路由、路由表、互联网网关、启动组、安全组、虚拟私有云、竞价型实例集和子网。
- elasticloadbalancing— 允许委托人创建、描述和删除 Elastic Load Balancing 负载平衡器。主体还能够向新创建的目标组、侦听器和负载均衡器的侦听器规则添加标签。
- events— 允许委托人创建、托管和删除 Amazon EventBridge 规则及其目标。
- iam— 允许委托人列出 IAM 角色及其附加的策略。委托人还可以列出 Amazon EC2 实例可用的实例配置文件。
- logs— 允许委托人创建和描述 Amazon CloudWatch Logs 日志组。委托人还可以列出这些日志组的日志事件。
- route53— 允许委托人创建、托管和删除 Amazon Route 53 托管区域。委托人还可以查看 Amazon Route 53 运行状况检查配置和信息。有关托管区更多信息，请参阅使用私有托管区域。
- servicediscovery— 允许委托人创建、管理和删除 AWS Cloud Map 服务并创建私有 DNS 命名空间。

以下是示例 AmazonECS_FullAccess 策略。

```
{
    "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",
                "appmesh:DescribeVirtualGateway",
                "appmesh:DescribeVirtualNode",
                "appmesh:ListMeshes",
                "appmesh:ListVirtualGateways",
                "appmesh:ListVirtualNodes",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:Describe*",
                "autoscaling:UpdateAutoScalingGroup",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetApplications",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:ContinueDeployment",
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetApplicationRevision",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:RegisterApplicationRevision",
```

```
                "codedeploy:StopDeployment",
                "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:RequestSpotFleet",
                "ec2:RunInstances",
                "ecs:*",
                "elasticfilesystem:DescribeAccessPoints",
                "elasticfilesystem:DescribeFileSystems",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateRule",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteRule",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:DescribeTargetGroups",
                "events:DeleteRule",
                "events:DescribeRule",
                "events:ListRuleNamesByTarget",
                "events:ListTargetsByRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
```

```
                "fsx:DescribeFileSystems",
                "iam:ListAttachedRolePolicies",
                "iam:ListInstanceProfiles",
                "iam:ListRoles",
                "lambda:ListFunctions",
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:FilterLogEvents",
                "route53:CreateHostedZone",
                "route53:DeleteHostedZone",
                "route53:GetHealthCheck",
                "route53:GetHostedZone",
                "route53:ListHostedZonesByName",
                "servicediscovery:CreatePrivateDnsNamespace",
                "servicediscovery:CreateService",
                "servicediscovery:DeleteService",
                "servicediscovery:GetNamespace",
                "servicediscovery:GetOperation",
                "servicediscovery:GetService",
                "servicediscovery:ListNamespaces",
                "servicediscovery:ListServices",
                "servicediscovery:UpdateService",
                "sns:ListTopics"
            ],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:GetParametersByPath"
            ],
            "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": [
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com",
                        "ecs.application-autoscaling.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": ["elasticloadbalancing:AddTags"],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticloadbalancing:CreateAction": [
                        "CreateTargetGroup",
                        "CreateRule",
                        "CreateListener",
                        "CreateLoadBalancer"
                    ]
                }
            }
        }
    ]
}
```

AmazonECSInfrastructureRolePolicyForVolumes

AmazonECSInfrastructureRolePolicyForVolumes 托管 IAM 策略授予 Amazon ECS 代表您进行 AWS API 调用所需的权限。您可以将此策略附加到启动 Amazon ECS 任务和服务时随卷配置提供的 IAM 角色。该角色使 Amazon ECS 能够管理附加到任务的卷。有关更多信息，请参阅 Amazon ECS 基础设施 IAM 角色。

权限详细信息

AmazonECSInfrastructureRolePolicyForVolumes 托管 IAM policy 必须包含以下权限。遵循授予最低权限的标准安全建议，您可以使用 AmazonECSInfrastructureRolePolicyForVolumes 托管策略作为创建仅包含所需权限的自定义策略的模板。

- ec2:CreateVolume – 当且仅当主体标有 AmazonECSCreated 和 AmazonECSManaged 标签时，才允许其创建 Amazon EBS 卷。需要此权限才能创建附加到 Amazon ECS 任务的 Amazon EBS 卷并最大限度地减少此策略向 Amazon ECS 提供的权限。
- ec2:CreateTags – 允许主体向 Amazon EBS 卷添加标签作为 ec2:CreateVolume 的一部分。Amazon ECS 需要此权限才能将客户指定的标签添加到代表您创建的 Amazon EBS 卷。
- ec2:AttachVolume – 允许主体将 Amazon EBS 卷附加到 Amazon EC2 实例。Amazon ECS 需要此权限才能将 Amazon EBS 卷附加到托管相关 Amazon ECS 任务的 Amazon EC2 实例。
- ec2:DescribeVolume – 允许主体检索有关 Amazon EBS 卷的信息。管理 Amazon EBS 卷的生命周期需要此权限。
- ec2:DescribeAvailabilityZones – 允许主体检索有关您账户中可用区的信息。这是管理 EBS 卷的生命周期所必需的。
- ec2:DetachVolume – 允许主体将 Amazon EBS 卷与 Amazon EC2 实例分离。任务终止时，Amazon ECS 需要此权限才能将 Amazon EBS 卷与托管相关 Amazon ECS 任务的 Amazon EC2 实例分离。
- ec2:DeleteVolume – 允许主体删除 Amazon EBS 卷。Amazon ECS 需要此权限才能删除 Amazon ECS 任务不再使用的 Amazon EBS 卷。
- ec2:DeleteTags – 允许主体从 Amazon EBS 卷中删除 AmazonECSManaged 标签。Amazon EBS 卷不再与 Amazon ECS 工作负载关联后，Amazon ECS 需要此权限才能删除对该卷的访问权限。这仅适用于任务关闭后未删除 Amazon EBS 卷的情况。

以下是示例 AmazonECSInfrastructureRolePolicyForVolumes 策略。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:CreateVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:RequestTag/AmazonECSManaged": "true"
    }
```

```
    }
   },
   {
    "Sid": "TagOnCreateVolume",
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "ArnLike": {
      "aws:RequestTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
     },
     "StringEquals": {
      "ec2:CreateAction": "CreateVolume",
      "aws:RequestTag/AmazonECSManaged": "true"
     }
    }
   },
   {
    "Sid": "DescribeVolumesForLifecycle",
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeVolumes",
     "ec2:DescribeAvailabilityZones"
    ],
    "Resource": "*"
   },
   {
    "Sid": "ManageEBSVolumeLifecycle",
    "Effect": "Allow",
    "Action": [
     "ec2:AttachVolume",
     "ec2:DetachVolume"
    ],
    "Resource": "arn:aws:ec2:*:*:volume/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/AmazonECSManaged": "true"
     }
    }
   },
   {
    "Sid": "ManageVolumeAttachmentsForEC2",
    "Effect": "Allow",
    "Action": [
```

```
    "ec2:AttachVolume",
    "ec2:DetachVolume"
   ],
   "Resource": "arn:aws:ec2:*:*:instance/*"
  },
  {
   "Sid": "DeleteEBSManagedVolume",
   "Effect": "Allow",
   "Action": "ec2:DeleteVolume",
   "Resource": "arn:aws:ec2:*:*:volume/*",
   "Condition": {
    "ArnLike": {
     "aws:ResourceTag/AmazonECSCreated": "arn:aws:ecs:*:*:task/*"
    },
    "StringEquals": {
     "aws:ResourceTag/AmazonECSManaged": "true"
    }
   }
  }
 ]
}
```

AmazonEC2ContainerServiceforEC2Role

Amazon ECS 将此策略附加到服务角色，该角色允许 Amazon ECS 代表您对 Amazon EC2 实例或外部实例执行操作。

此策略授予允许Amazon ECS容器实例代表您调用 AWS 的管理权限。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。

注意事项

您应注意以下建议和注意事项，使用 AmazonEC2ContainerServiceforEC2Role 托管 IAM policy。

- 遵循授予最低权限的标准安全建议，您可以修改 AmazonEC2ContainerServiceforEC2Role 托管策略，以满足您的特定需求。如果您的用例不需要托管策略中授予的任何权限，请创建自定义策略并仅添加所需的权限。例如，为 Spot Instance 耗尽提供 UpdateContainerInstancesState 权限。如果您的用例不需要该权限，请使用自定义策略将其排除。有关更多信息，请参阅 权限详细信息。
- 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 Amazon ECS 任务 IAM 角色。

您可以防止在 `docker0` 网桥访问提供给容器实例角色的权限。您可以在仍然允许通过在容器实例上运行以下命令 Amazon ECS 任务 IAM 角色 提供的权限的情况下执行 iptables 此操作。容器无法查询此规则生效的实例元数据。此命令采用原定设置 Docker 网桥配置，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。

```
sudo yum install -y iptables-services; sudo iptables --insert DOCKER USER 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。对于经 Amazon ECS 优化的 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。

- 对于经 Amazon ECS 优化的 Amazon Linux 2 AMI：

  ```
  sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now iptables
  ```

- 对于经 Amazon ECS 优化的 Amazon Linux AMI：

  ```
  sudo service iptables save
  ```

权限详细信息

`AmazonEC2ContainerServiceforEC2Role` 托管 IAM policy 必须包含以下权限。遵循授予最低权限的标准安全建议，`AmazonEC2ContainerServiceforEC2Role` 托管策略可用作指南。如果不需要在托管策略中为您的用例授予的任何权限，请创建自定义策略并仅添加所需的权限。

- `ec2:DescribeTags`— 允许委托人描述与 Amazon EC2 实例关联的标签 Amazon ECS 容器代理使用此权限来支持资源标签传播。有关更多信息，请参阅 如何为资源添加标签。
- `ecs:CreateCluster`— 允许委托人创建 Amazon ECS 集群。Amazon ECS 容器代理将使用此权限创建 `default` 集群（如果还没有集群）。
- `ecs:DeregisterContainerInstance`— 允许委托人从集群注销 Amazon ECS 容器实例。Amazon ECS 容器代理不会调用此 API 操作，但仍保留此权限以帮助确保后续的兼容性。
- `ecs:DiscoverPollEndpoint`— 此操作返回 Amazon ECS 容器代理用于轮询更新的端点。
- `ecs:Poll`— 允许 Amazon ECS 容器代理与 Amazon ECS 控制平面通信，报告任务状态更改。

- ecs:RegisterContainerInstance— 允许委托人向集群注册容器实例。Amazon ECS 容器代理使用此权限向集群注册 Amazon EC2 实例并支持资源标签传播。
- ecs:StartTelemetrySession— 允许 Amazon ECS 容器代理与 Amazon ECS 控制平面通信，报告每个容器和任务的运行状况信息和指标。
- ecs:TagResource – 允许 Amazon ECS 容器代理在创建集群时标记集群，并在容器实例注册到集群时对其进行标记。
- ecs:UpdateContainerInstancesState— 允许委托人修改 Amazon ECS 容器实例的状态。Amazon ECS 容器代理将此权限用于 Spot Instance 耗尽。
- ecs:Submit* –这包括 SubmitAttachmentStateChanges、SubmitContainerStateChange 和 SubmitTaskStateChange API 操作。Amazon ECS 容器代理使用它们向 Amazon ECS 控制平面报告每个资源的状态变化。SubmitContainerStateChange 权限不再被 Amazon ECS 容器代理使用，但仍可帮助确保后续的兼容性。
- ecr:GetAuthorizationToken— 允许委托人检索授权令牌。授权令牌表示您的 IAM 身份验证凭证，可用于访问您的 IAM 委托人有权访问的任何 Amazon ECR 注册表。收到的授权令牌有效期为 12 小时。
- ecr:BatchCheckLayerAvailability— 将容器映像推送到 Amazon ECR 私有存储库时，会检查每个映像层验证它是否已推送。如果是，则会跳过映像层。
- ecr:GetDownloadUrlForLayer— 从 Amazon ECR 专用存储库中提取容器映像时，对于尚未缓存的每个图像图层，此 API 都会调用一次。
- ecr:BatchGetImage— 从 Amazon ECR 私有存储库提取容器映像时，会调用一次此 API 以检索映像清单。
- logs:CreateLogStream— 允许委托人为指定的日志组创建 CloudWatch Logs 日志流。
- logs:PutLogEvents— 允许委托人将一批日志事件上传到指定的日志流。

以下是示例 AmazonEC2ContainerServiceforEC2Role 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
```

```
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "CreateCluster",
                        "RegisterContainerInstance"
                    ]
                }
            }
        }
    ]
}
```

AmazonEC2ContainerServiceEventsRole

此策略授予允许 Amazon EventBridge（以前的 CloudWatch Events）代表您运行任务的权限。此策略可附加到创建计划任务时指定的 IAM 角色。有关更多信息，请参阅 Amazon ECS EventBridge IAM 角色。

权限详细信息

该策略包含以下权限。

- ecs— 允许服务中的委托人调用 Amazon ECS RunTask API。允许服务中的主体在调用 Amazon ECS RunTask API 时添加标签（TagResource）。

- `iam`— 允许将任何 IAM 服务角色传递给任何 Amazon ECS 任务。

以下是示例 `AmazonEC2ContainerServiceEventsRole` 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

AmazonECSTaskExecutionRolePolicy

`AmazonECSTaskExecutionRolePolicy` 托管 IAM policy 授予 Amazon ECS 容器代理和 AWS Fargate 要创建的容器代理代表您调用 AWS API。此策略可添加到您的任务执行 IAM 角色中。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

权限详细信息

AmazonECSTaskExecutionRolePolicy 托管 IAM policy 必须包含以下权限。遵循授予最低权限的标准安全建议，AmazonECSTaskExecutionRolePolicy 托管策略可用作指南。如果您的用例不需要托管策略中授予的任何权限，请创建自定义策略并仅添加所需的权限。

- ecr:GetAuthorizationToken— 允许委托人检索授权令牌。授权令牌表示您的 IAM 身份验证凭证，可用于访问您的 IAM 委托人有权访问的任何 Amazon ECR 注册表。收到的授权令牌有效期为 12 小时。
- ecr:BatchCheckLayerAvailability— 将容器映像推送到 Amazon ECR 私有存储库时，会检查每个映像层验证它是否已推送。如果已推送，则会跳过映像层。
- ecr:GetDownloadUrlForLayer— 从 Amazon ECR 专用存储库中提取容器映像时，对于尚未缓存的每个图像图层，此 API 都会调用一次。
- ecr:BatchGetImage— 从 Amazon ECR 私有存储库提取容器映像时，会调用一次此 API 以检索映像清单。
- logs:CreateLogStream— 允许委托人为指定的日志组创建 CloudWatch Logs 日志流。
- logs:PutLogEvents— 允许委托人将一批日志事件上传到指定的日志流。

以下是示例 AmazonECSTaskExecutionRolePolicy 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECSServiceRolePolicy

AmazonECSServiceRolePolicy 托管 IAM policy 使 Amazon Elastic Container Service 能够管理您的集群。此策略可添加到您的任务执行 IAM 角色中。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

权限详细信息

AmazonECSServiceRolePolicy 托管 IAM policy 必须包含以下权限。遵循授予最低权限的标准安全建议，AmazonECSServiceRolePolicy 托管策略可用作指南。如果您的用例不需要托管策略中授予的任何权限，请创建自定义策略并仅添加所需的权限。

- autoscaling— 允许委托人创建、托管和描述 Amazon EC2 Auto Scaling 资源。使用集群自动扩缩功能时，在管理 Amazon EC2 Auto Scaling 组时需要此项。
- autoscaling-plans – 允许主体创建、删除和描述自动扩缩计划。
- cloudwatch— 允许委托人创建、托管和描述 Amazon CloudWatch 警报。
- ec2 – 允许主体运行到 Amazon EC2 实例及创建和管理网络接口和标签。
- elasticloadbalancing— 允许委托人创建、描述和删除 Elastic Load Balancing 负载平衡器。主体还将能够添加和描述目标组。
- logs— 允许委托人创建和描述 Amazon CloudWatch Logs 日志组。委托人还可以列出这些日志组的日志事件。
- route53— 允许委托人创建、托管和删除 Amazon Route 53 托管区域。委托人还可以查看 Amazon Route 53 运行状况检查配置和信息。有关托管区更多信息，请参阅使用私有托管区域。
- servicediscovery— 允许委托人创建、管理和删除 AWS Cloud Map 服务并创建私有 DNS 命名空间。
- events— 允许委托人创建、托管和删除 Amazon EventBridge 规则及其目标。

以下是示例 AmazonECSServiceRolePolicy 策略。

```
{
    "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": "AutoScaling",
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AutoScalingManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DeletePolicy",
                "autoscaling:PutScalingPolicy",
                "autoscaling:SetInstanceProtection",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:PutLifecycleHook",
                "autoscaling:DeleteLifecycleHook",
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:RecordLifecycleActionHeartbeat"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "autoscaling:ResourceTag/AmazonECSManaged": "false"
                }
            }
        },
        {
            "Sid": "AutoScalingPlanManagement",
            "Effect": "Allow",
            "Action": [
                "autoscaling-plans:CreateScalingPlan",
                "autoscaling-plans:DeleteScalingPlan",
                "autoscaling-plans:DescribeScalingPlans",
                "autoscaling-plans:DescribeScalingPlanResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EventBridge",
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:ListTargetsByRule"
            ],
            "Resource": "arn:aws:events:*:*:rule/ecs-managed-*"
        },
        {
            "Sid": "EventBridgeRuleManagement",
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:PutTargets"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "events:ManagedBy": "ecs.amazonaws.com"
                }
            }
        },
        {
            "Sid": "CWAlarmManagement",
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Sid": "ECSTagging",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Sid": "CWLogGroupManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*"
        },
        {
            "Sid": "CWLogStreamManagement",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/ecs/*:log-stream:*"
        },
        {
            "Sid": "ExecuteCommandSessionManagement",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "ExecuteCommand",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ecs:*:*:task/*",
                "arn:aws:ssm:*:*:document/AmazonECS-ExecuteInteractiveCommand"
            ]
        },
        {
            "Sid": "CloudMapResourceCreation",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:CreateHttpNamespace",
                "servicediscovery:CreateService"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "AmazonECSManaged"
                    ]
                }
            }
        },
        {
            "Sid": "CloudMapResourceTagging",
            "Effect": "Allow",
            "Action": "servicediscovery:TagResource",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/AmazonECSManaged": "*"
                }
            }
        },
        {
            "Sid": "CloudMapResourceDeletion",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DeleteService"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonECSManaged": "false"
                }
            }
        },
        {
            "Sid": "CloudMapResourceDiscovery",
            "Effect": "Allow",
            "Action": [
                "servicediscovery:DiscoverInstances",
                "servicediscovery:DiscoverInstancesRevision"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecur

提供代表您管理 Amazon ECS Service Connect TLS 功能所需的对 AWS Private Certificate Authority、Secrets Manager 和其他 AWS 服务的管理权限。

权限详细信息

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 托管 IAM policy 必须包含以下权限。遵循授予最低权限的标准安全建议，AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 托管策略可用作指南。如果您的用例不需要托管策略中授予的任何权限，请创建自定义策略并仅添加所需的权限。

- secretsmanager:CreateSecret – 允许主体创建密钥。Service Connect TLS 需要此权限，这样 Amazon ECS 才能将客户的私有密钥保存在客户的 Secrets Manager 密钥中。
- secretsmanager:TagResource – 允许主体在创建的密钥上附加标签。Service Connect TLS 需要此权限，因为 Amazon ECS 代表客户创建密钥并将标签附加到资源。这些标签为客户提供了一种更简单的方法，以识别托管密钥并限制对这些密钥的操作。
- secretsmanager:DescribeSecret – 允许主体描述密钥并检索当前版本阶段。Amazon ECS 需要进行 Amazon ECS Service Connect TLS 材料轮换。

- secretsmanager:UpdateSecret – 允许主体更新密钥。Amazon ECS 需要进行 Amazon ECS Service Connect TLS 材料轮换并使用新材料更新密钥。
- secretsmanager:GetSecretValue – 允许主体获取密钥值。Amazon ECS 需要进行 Amazon ECS Service Connect TLS 材料轮换。
- secretsmanager:PutSecretValue – 允许主体放入密钥值。Amazon ECS 需要进行 Amazon ECS Service Connect TLS 材料轮换。
- secretsmanager:UpdateSecretVersionStage – 允许主体更新密钥版本阶段。Amazon ECS 需要进行 Amazon ECS Service Connect TLS 材料轮换。
- acm-pca:IssueCertificate – 允许主体为 Amazon ECS Service Connect TLS 的 End entity certificate 调用 IssueCertificate。ECS 需要此权限为客户的上游服务生成证书。
- acm-pca:GetCertificate – 允许主体为 Amazon ECS Service Connect TLS 的 End entity certificate 调用 GetCertificate。
- acm-pca:GetCertificateAuthorityCertificate – 允许主体获取证书颁发机构证书。Amazon ECS Service Connect TLS 需要此权限，这样客户的下游服务才能信任上游终端实体证书。
- acm-pca:DescribeCertificateAuthority – 允许主体获取有关证书颁发机构的详细信息。Amazon ECS Service Connect TLS 需要此权限才能重复使用诸如签名算法之类的信息来创建 CSR（证书签名请求）。

以下是示例 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:CreateSecret",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                    "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
```

```
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "TagOnCreateSecret",
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "ArnLike": {
                     "aws:RequestTag/AmazonECSCreated": [
                        "arn:aws:ecs:*:*:service/*/*",
                        "arn:aws:ecs:*:*:task-set/*/*"
                    ]
                },
                "StringEquals": {
                    "aws:RequestTag/AmazonECSManaged": "true",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "RotateTLSCertificateSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:UpdateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:RotateSecret",
                "secretsmanager:UpdateSecretVersionStage"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
"ecs-sc",
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
```

```
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthority",
            "Effect": "Allow",
            "Action": [
                "acm-pca:GetCertificate",
                "acm-pca:GetCertificateAuthorityCertificate",
                "acm-pca:DescribeCertificateAuthority"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true"
                }
            }
        },
        {
            "Sid": "ManagePrivateCertificateAuthorityForIssuingEndEntityCertificate",
            "Effect": "Allow",
            "Action": [
                "acm-pca:IssueCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/AmazonECSManaged": "true",
                    "acm-pca:TemplateArn":"arn:aws:acm-pca:::template/
EndEntityCertificate/V1"
                }
            }
        }
    ]
}
```

AWSApplicationAutoscalingECSServicePolicy

您不能将 AWSApplicationAutoscalingECSServicePolicy 附加到自己的 IAM 实体。此策略附加到服务链接角色，该角色允许 Application Auto Scaling 以代表您执行操作。有关更多信息，请参阅 Application Auto Scaling 服务相关角色。

AWSCodeDeployRoleForECS

您不能将 AWSCodeDeployRoleForECS 附加到自己的 IAM 实体。此策略附加到服务链接角色，该角色允许 CodeDeploy 代表您执行操作。有关更多信息，请参阅 AWS CodeDeploy 用户指南中的为 CodeDeploy 创建服务角色。

AWSCodeDeployRoleForECSLimited

您不能将 AWSCodeDeployRoleForECSLimited 附加到自己的 IAM 实体。此策略附加到服务链接角色，该角色允许 CodeDeploy 代表您执行操作。有关更多信息，请参阅 AWS CodeDeploy 用户指南中的为 CodeDeploy 创建服务角色。

Amazon ECS 更新为 AWS 托管策略

查看自此服务开始跟踪这些更改以来 Amazon ECS AWS 托管策略更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon ECS 文档历史记录页面上的 RSS 源。

更改	描述	日期
新增 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 策略	新增 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 策略，该策略提供对 AWS KMS、AWS Private Certificate Authority、Secrets Manager 的管理访问权限，并使 Amazon ECS Service Connect TLS 功能能够正常工作。	2024 年 1 月 22 日
新增策略 AmazonECSInfrastructureRolePolicyForVolumes	已添加 AmazonECSInfrastructureRolePolicyForVolumes 策略。该策略授予 Amazon ECS 进行 AWS API 调用所需的权限，以管理与 Amazon ECS 工作负载关联的 Amazon EBS 卷。	2024 年 1 月 11 日

更改	描述	日期
将权限添加到 AmazonECS ServiceRolePolicy	AmazonECSServiceRolePolicy 托管 IAM 策略已更新，增加了新的 events 权限，以及附加的 autoscaling 和 autoscaling-plans 权限。	2023 年 12 月 4 日
将权限添加到 AmazonEC2 ContainerServiceEventsRole	AmazonECSServiceRolePolicy 托管 IAM 策略已更新，允许访问 AWS Cloud Map DiscoverInstancesRevision API 操作。	2023 年 10 月 4 日
将权限添加到 AmazonEC2 ContainerServiceforEC2Role	修改 AmazonEC2ContainerServiceforEC2Role 策略是为了添加 ecs:TagResource 权限，其中包括一个条件，该条件将权限限制为仅限于新创建的集群和注册的容器实例。	2023 年 3 月 6 日
向 the section called “AmazonECS_FullAccess” 添加权限	修改 AmazonECS_FullAccess 策略是为了添加 elasticloadbalancing:AddTags 权限，其中包括一个条件，该条件将权限限制为仅限于新创建的负载均衡器、目标组、规则和创建的侦听器。此权限不允许向任何已经创建的 Elastic Load Balancing 资源添加标签。	2023 年 1 月 4 日
Amazon ECS 开始跟踪更改	Amazon ECS 开始跟踪其 AWS 托管策略的更改。	2021 年 6 月 8 日

逐步淘汰 Amazon Elastic Container Service 的 AWS 托管 IAM policy

以下 AWS 托管 IAM policy 将逐步停用。这些策略现在已由更新的策略取代。我们建议您更新用户或角色，来使用更新的策略。

AmazonEC2ContainerServiceFullAccess

> **Important**
>
> `AmazonEC2ContainerServiceFullAccess` 托管 IAM policy 已于 2021 年 1 月 29 日逐步停用，以响应 `iam:passRole` 权限内的安全问题。此权限授予对所有资源的访问权限，包括账户中角色的凭证。现在该策略已逐步停用，您无法将该策略附加到任何新的用户或角色。任何已附加策略的用户或角色都可以继续使用它。但是，我们建议您更新用户或角色，而不是使用 `AmazonECS_FullAccess` 托管策略。有关更多信息，请参阅 迁移到 `AmazonECS_FullAccess` 托管策略。

AmazonEC2ContainerServiceRole

> **Important**
>
> `AmazonEC2ContainerServiceRole` 托管 IAM policy 已逐步停用。现在已被 Amazon ECS 与 Amazon ECS 服务相关角色所取代。有关更多信息，请参阅 对 Amazon ECS 使用服务相关角色。

AmazonEC2ContainerServiceAutoscaleRole

> **Important**
>
> `AmazonEC2ContainerServiceAutoscaleRole` 托管 IAM policy 已逐步停用。现在，已被 Amazon ECS 的 Application Auto Scaling 服务链接角色所取代。有关更多信息，请参阅《Application Auto Scaling 用户指南》中的 Application Auto Scaling 服务相关角色。

迁移到 `AmazonECS_FullAccess` 托管策略

`AmazonEC2ContainerServiceFullAccess` 托管 IAM policy 已于 2021 年 1 月 29 日逐步停用，以响应 `iam:passRole` 权限内的安全问题。此权限授予对所有资源的访问权限，包括账户中

角色的凭证。现在该策略已逐步停用，您无法将该策略附加到任何新的组、用户或角色。任何已附加策略的组、用户或角色都可以继续使用它。但是，我们建议您更新组、用户或角色，以使用 AmazonECS_FullAccess 托管策略。

授予的权限由 AmazonECS_FullAccess 策略包括将 ECS 用作管理员所需的完整权限列表。如果您当前使用的由 AmazonEC2ContainerServiceFullAccess 策略授予的权限中不在 AmazonECS_FullAccess 策略中，您可以将其添加到内联策略语句中。有关更多信息，请参阅 Amazon Elastic Container Service AWS 托管策略。

使用以下步骤确定您是否有任何组、用户或角色当前正在使用 AmazonEC2ContainerServiceFullAccess 托管 IAM policy。然后，更新它们以分离早期的策略并附加 AmazonECS_FullAccess 政策。

更新组、用户或角色以使用 AmazonECS_FullAccess 策略 (AWS Management Console)

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择策略并搜索并选择 AmazonEC2ContainerServiceFullAccess 政策。
3. 选择策略用法选项卡，显示当前正在使用此策略的任何 IAM 角色。
4. 对于当前使用 AmazonEC2ContainerServiceFullAccess 策略的 IAM 角色，请选择角色，然后使用以下步骤分离逐步停用的策略并附加 AmazonECS_FullAccess 策略。

    a. 在 Permissions (权限)选项卡上，选择 AmazonEC2ContainerServiceFullAccess 策略旁边的 X。
    b. 选择添加权限。
    c. 选择直接附加现有策略中，搜索并选择 AmazonECS_FullAccess 策略，然后选择下一步:审核。
    d. 查看更改，然后选择添加权限。
    e. 对使用 AmazonEC2ContainerServiceFullAccess 策略的每个组、用户或角色重复这些步骤。

更新组、用户或角色以使用 **AmazonECS_FullAccess** 策略 (AWS CLI)

1. 使用 generate-service-last-accessed-details 命令生成报告，其中包含有关上次使用逐步停用策略时的详细信息。

```
aws iam generate-service-last-accessed-details \
     --arn arn:aws:iam::aws:policy/AmazonEC2ContainerServiceFullAccess
```

输出示例：

```
{
    "JobId": "32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE"
}
```

2. 将先前输出中的作业 ID 与 get-service-last-accessed-details 命令配合使用以检索服务的上次访问报告。此报告显示上次使用逐步停用策略的 IAM 实体的 Amazon 资源名称（ARN）。

```
aws iam get-service-last-accessed-details \
      --job-id 32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE
```

3. 使用以下命令之一将 AmazonEC2ContainerServiceFullAccess 策略与组、用户或角色分离。

   - detach-group-policy
   - detach-role-policy
   - detach-user-policy

4. 使用以下命令之一将 AmazonECS_FullAccess 策略附加到组、用户或角色。

   - attach-group-policy
   - attach-role-policy
   - attach-user-policy

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独特类型的 IAM 角色，它与 Amazon ECS 直接相关。服务相关角色是由 Amazon ECS 预定义的，并包含服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon ECS，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，除非另外定义，否则只有 Amazon ECS 可以代入该角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其它 IAM 实体。

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，并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务相关角色文档。

Amazon ECS 的服务相关角色权限

Amazon ECS 使用名为 AWSServiceRoleForECS 的服务相关角色。

AWSServiceRoleForECS 服务相关角色信任以下服务以担任该角色：

- `ecs.amazonaws.com`

名为 AmazonECSServiceRolePolicy 的角色权限策略允许 Amazon ECS 对指定资源完成以下操作：

- 操作：针对您的 Amazon ECS 任务使用 `awsvpc` 网络模式时，Amazon ECS 会管理与任务关联的弹性网络接口的生命周期。这还包括 Amazon ECS 向您的弹性网络接口添加的标签。
- 操作：通过 Amazon ECS 服务使用负载均衡器时，Amazon ECS 会管理负载均衡器中资源的注册和注销。
- 操作：使用 Amazon ECS 服务发现时，Amazon ECS 会管理所需的 Route 53 和 AWS Cloud Map 资源，使服务发现能正常运作。
- 操作：使用 Amazon ECS 服务自动扩缩时，Amazon ECS 会管理所需的自动扩缩资源。
- 操作：Amazon ECS 会创建和管理 CloudWatch 警报和日志流，以帮助监控您的 Amazon ECS 资源。
- 操作：使用 Amazon ECS Exec 时，Amazon ECS 会管理启动任务 Amazon ECS Exec 会话所需的权限。
- 操作：使用 Amazon ECS Service Connect 时，Amazon ECS 会管理使用该功能所需的 AWS Cloud Map 资源。
- 操作：使用 Amazon ECS 容量提供程序时，Amazon ECS 会管理修改自动扩缩组及其 Amazon EC2 实例所需的权限。

您必须配置权限，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。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

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。当您创建集群或者在 AWS Management Console、AWS CLI 或 AWS API 中创建或更新服务时，Amazon ECS 会为您创建服务相关角色。如果您在创建集群后看不到 AWSServiceRoleForECS 角色，请执行以下操作来修复问题：

- 验证并配置权限以允许 Amazon ECS 代表您创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

- 重试集群创建操作，或手动创建服务相关角色。

  您可以使用 IAM 控制台创建 AWSServiceRoleForECS 服务相关角色。在 AWS CLI 或 AWS API 中，使用 ecs.amazonaws.com 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您创建集群或者创建或更新服务时，Amazon ECS 会再次为您创建服务相关角色。

如果删除该服务相关角色，您可以使用相同的 IAM 过程再次创建该角色。

为 Amazon ECS 编辑服务相关角色

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。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除适用于 Amazon ECS 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果当您试图删除资源时 Amazon ECS 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

检查服务相关角色是否有活动会话

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择Roles（角色）并选择 AWSServiceRoleForECS 名称（而不是复选框）。

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。

要删除 AWSServiceRoleForECS 服务链接角色使用的 Amazon ECS 资源

在删除 AWSServiceRoleForECS 角色之前，必须先删除所有 AWS 区域中的所有 Amazon ECS 群集。

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，请参阅使用控制台更新 Amazon ECS 服务 和使用控制台删除 Amazon ECS 服务。
2. 强制从所有区域的所有集群中注销所有容器实例。有关更多信息，请参阅 注销 Amazon ECS 容器实例。
3. 删除所有区域的所有 Amazon ECS 群集。有关更多信息，请参阅 删除 Amazon ECS 集群。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 删除 AWSServiceRoleForECS 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

Amazon ECS 服务相关角色支持的区域

Amazon ECS 支持在该服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和端点。

适用于 Amazon ECS 的 IAM 角色

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。在 Amazon ECS 中，您可以创建角色来授予对 Amazon ECS 资源（例如容器或服务）的权限。

Amazon ECS 需要的角色取决于任务定义启动类型和您使用的功能。使用下表确定 Amazon ECS 需要哪些 IAM 角色。

角色	定义	何时需要	更多信息
任务执行角色	此角色允许 Amazon ECS 代表您使用其他 AWS 服务。	您的任务托管在 AWS Fargate 或外部实例上且： • 从 Amazon ECR 私有存储库中提取容器映像。 • 在与运行任务的账户不同的账户中，从 Amazon ECR 私有存储库中提取容器映像。 • 使用 `awslogs` 日志驱动程序将容器日志发送到 CloudWatch Logs。 您的任务托管在 AWS Fargate 或 Amazon EC2 实例上且： • 使用私有注册表身份验证。 • 使用运行时监控。 • 任务定义使用 Secrets Manager 密钥或 AWS Systems Manager Parameter Store 参数引用敏感数据。	Amazon ECS 任务执行 IAM 角色
任务角色	此角色允许您的应用程序代码（在容器上	您的应用程序访问其他 AWS 服务，例如 Amazon S3。	Amazon ECS 任务 IAM 角色

角色	定义	何时需要	更多信息
	）使用其他 AWS 服务。		
容器实例角色	此角色允许您的 EC2 实例或外部实例向集群注册。	您的任务托管在 Amazon EC2 实例或外部实例上。	Amazon ECS 容器实例 IAM 角色
Amazon ECS Anywhere 角色	此角色允许您的外部实例访问 AWS API。	您的任务托管在外部实例上。	Amazon ECS Anywhere IAM 角色
Amazon ECS CodeDeploy 角色	此角色允许 CodeDeploy 对您的服务进行更新。	您可使用 CodeDeploy 蓝/绿部署类型来部署服务。	Amazon ECS CodeDeploy IAM 角色
Amazon ECS EventBridge 角色	此角色允许 EventBridge 对您的服务进行更新。	您可使用 EventBridge 规则和目标来计划任务。	Amazon ECS EventBridge IAM 角色
Amazon ECS 基础设施角色	此角色允许 Amazon ECS 管理集群中的基础设施资源。	• 您想将 Amazon EBS 卷附加到您的 Fargate 或 EC2 启动类型 Amazon ECS 任务。基础设施角色允许 Amazon ECS 为您的任务管理 Amazon EBS 卷。 • 您想使用传输层安全性协议（TLS）加密您的 Amazon ECS Service Connect 服务之间的流量。	Amazon ECS 基础设施 IAM 角色

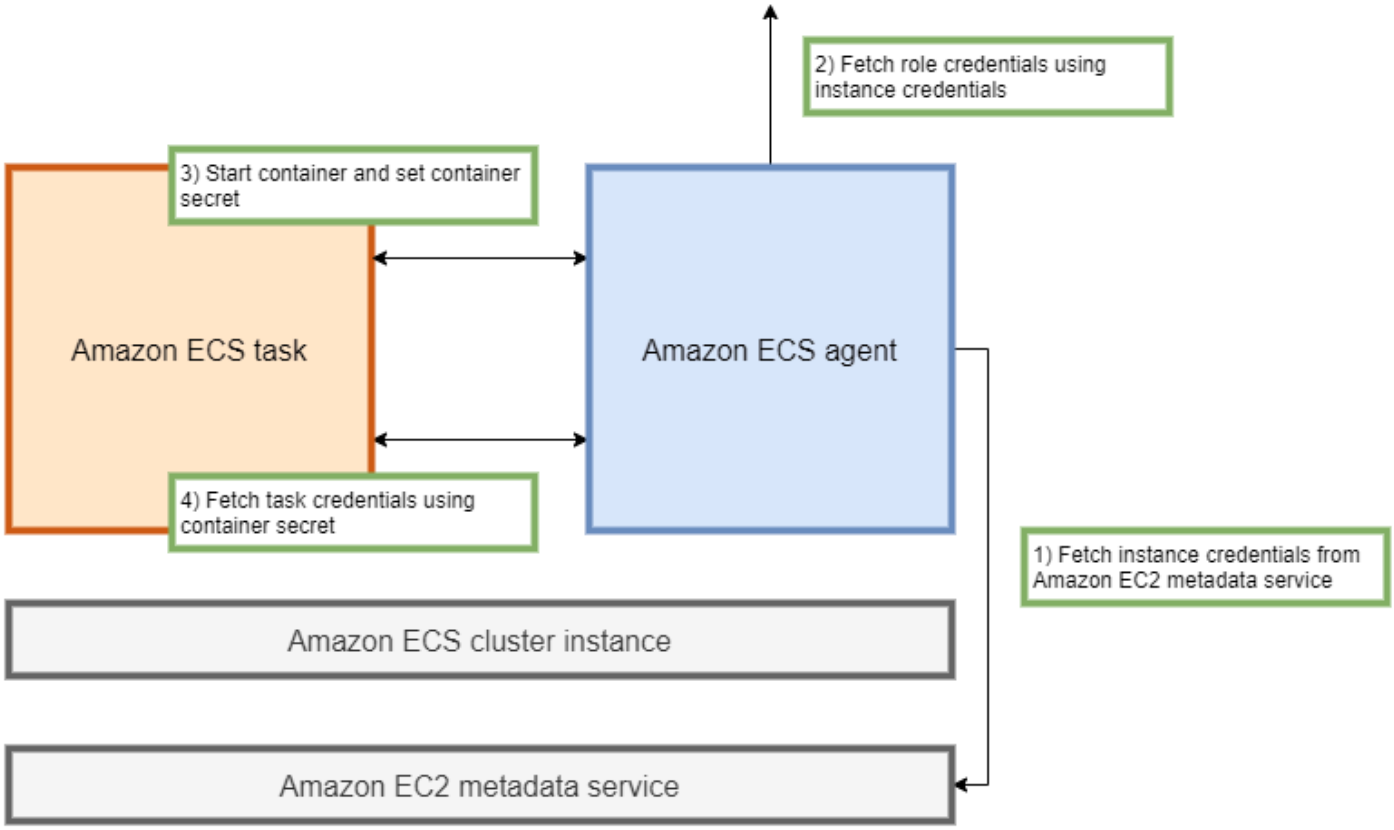
Amazon EC2 中 IAM 角色的最佳实践

建议您为任务分配角色。它的角色可以与其运行的 Amazon EC2 实例的角色不同。为每项任务分配角色符合最低权限访问的原则，并有利于对操作和资源进行更精细的控制。

在为任务分配 IAM 角色时，您必须使用以下信任策略，以便每个任务都可以承担与 EC2 实例使用的不同的 IAM 角色。这样，您的任务不会继承 EC2 实例的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

当您在任务定义中添加任务角色时，Amazon ECS 容器代理会自动为该任务创建具有唯一凭证 ID（例如 12345678-90ab-cdef-1234-567890abcdef）的令牌。然后，该令牌和角色凭证将添加到代理的内部缓存中。代理使用凭证 ID 的 URI 填充容器 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 中的环境变量（例如，/v2/credentials/12345678-90ab-cdef-1234-567890abcdef）。



您可以通过将环境变量附加到 Amazon ECS 容器代理的 IP 地址并对生成的字符串运行 `curl` 命令来手动检索容器内部的临时角色凭证。

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

预期的输出如下所示：

```
{
 "RoleArn": "arn:aws:iam::123456789012:role/SSMTaskRole-SSMFargateTaskIAMRole-
DASWWSF2WGD6",
 "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
 "Token": "IQoJb3JpZ2luX2VjEEM/Example==",
 "Expiration": "2021-01-16T00:51:53Z"
}
```

调用 AWS API 时，较新版本的 AWS 开发工具包会自动从 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 环境变量中获取这些凭证。

输出包括一个访问密钥对，该密钥对由您的应用程序用于访问 AWS 资源的秘密访问密钥 ID 和秘密密钥组成。它还包括 AWS 用于验证凭证是否有效的令牌。默认情况下，使用任务角色分配给任务的凭证有效期为六小时。之后，Amazon ECS 容器代理会自动轮换它们。

任务执行角色

任务执行角色用于授予 Amazon ECS 容器代理代表您调用特定 AWS API 操作的权限。例如，当您使用 AWS Fargate 时，Fargate 需要一个 IAM 角色来允许它从 Amazon ECR 提取映像并将日志写入 CloudWatch Logs。当任务引用存储在 AWS Secrets Manager 中的密钥（例如映像提取密钥）时，还需要一个 IAM 角色。

> **Note**
>
> 如果您以经过身份验证的用户身份提取映像，则不太可能受到 Docker Hub 拉取速率限制变化的影响。有关更多信息，请参阅容器实例的私有注册表身份验证。
> 通过使用 Amazon ECR 和 Amazon ECR Public，您可以避开 Docker 施加的限制。如果您从 Amazon ECR 提取映像，这还有助于缩短网络提取时间，并减少流量离开您的 VPC 时的数据传输变化。

> **Important**
>
> 使用 Fargate 时，您必须使用 `repositoryCredentials` 向私有映像注册表进行身份验证。无法为 Fargate 上托管的任务设置 Amazon ECS 容器代理环境变量 `ECS_ENGINE_AUTH_TYPE` 或 `ECS_ENGINE_AUTH_DATA` 或修改 `ecs.config` 文件。有关更多信息，请参阅任务的私有注册表身份验证。

容器实例角色

Amazon ECS 容器代理是在 Amazon ECS 集群内的每个 Amazon EC2 实例上运行的容器。它是使用操作系统上提供的 `init` 命令在 Amazon ECS 外部初始化的。因此，无法通过任务角色向其授予权限。相反，必须将权限分配给代理运行所在的 Amazon EC2 实例。示例 `AmazonEC2ContainerServiceforEC2Role` 策略中的操作列表需要授予给 `ecsInstanceRole`。如果您不这样做，您的实例将无法加入集群。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ecs:CreateCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:StartTelemetrySession",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

在本策略中，`ecr` 和 `logs` API 操作允许在您的实例上运行的容器从 Amazon ECR 提取映像并将日志写入 Amazon CloudWatch。`ecs` 操作允许代理注册和注销实例，以及与 Amazon ECS 控制面板进行通信。其中，`ecs:CreateCluster` 操作是可选的。

服务相关角色

您可以将服务相关角色用于 Amazon ECS，以授予 Amazon ECS 服务代表您调用其他服务 API 的权限。Amazon ECS 需要拥有创建和删除网络接口、向目标组注册和注销目标的权限。它还需要必要的权限才能创建和删除扩展策略。这些权限通过服务相关角色授予。此角色是在您首次使用该服务时代表您创建的。

> **Note**
>
> 如果您无意中删除了服务相关角色，则可以重新创建它。有关说明，请参阅创建服务相关角色。

角色建议

在设置任务 IAM 角色和策略时，建议执行以下操作。

阻止对 Amazon EC2 元数据的访问

当您在 Amazon EC2 实例上运行任务时，强烈建议您阻止访问 Amazon EC2 元数据，以防止您的容器继承分配给这些实例的角色。如果您的应用程序必须调用 AWS API 操作，请改用 IAM 角色执行任务。

要防止在桥接模式下运行的任务访问 Amazon EC2 元数据，请运行以下命令或更新实例的用户数据。有关更新实例用户数据的更多说明，请参阅此 AWS 支持文章。有关任务定义桥接模式的更多信息，请参阅任务定义网络模式。

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 192.0.2.0/32 --jump DROP
```

要使此更改在重启后保持不变，请运行以下特定于您的亚马逊机器映像（AMI）的命令：

- Amazon Linux 2

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

- Amazon Linux

```
sudo service iptables save
```

对于使用 awsvpc 网络模式的任务，请在 /etc/ecs/ecs.config 文件中将环境变量 ECS_AWSVPC_BLOCK_IMDS 设置为 true。

您应在 ecs-agent config 文件中将 ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST 变量设置为 false，以防止在 host 网络中运行的容器访问 Amazon EC2 元数据。

使用 **awsvpc** 网络模式

使用网络 awsvpc 网络模式限制不同任务之间或您的任务与在 Amazon VPC 内运行的其他服务之间的流量。这将额外增加一层安全性。awsvpc 网络模式为在 Amazon EC2 上运行的任务提供任务级别的网络隔离。它是 AWS Fargate 上的默认模式，是唯一可以用来为任务分配安全组的网络模式。

使用 IAM Access Advisor 来完善角色

建议您删除任何从未使用或一段时间未使用的操作。这样可以防止发生不必要的访问。为此，请查看 IAM Access Advisor 生成的结果，然后移除从未使用过或最近未使用过的操作。您可以按照以下步骤执行此操作。

运行以下命令以生成报告，其中显示所引用策略的最后访问信息：

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

使用输出中的 `JobId` 来运行以下命令。执行此操作后，您可以查看报告的结果。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

有关更多信息，请参阅 IAM Access Advisor。

监控 AWS CloudTrail 是否有可疑活动

您可以监控 AWS CloudTrail 是否有任何可疑活动。大多数 AWS API 调用都以事件形式记录到 AWS CloudTrail 中。AWS CloudTrail Insights 会对它们进行分析，并提醒您注意任何与 `write` API 调用相关的可疑行为。这可能包括通话量激增。这些警报包括异常活动发生的时间和促成 API 的高级身份 ARN 等信息。

您可以通过查看事件的 `userIdentity` 属性来识别具有 IAM 角色的任务在 AWS CloudTrail 中执行的操作。在以下示例中，`arn` 包括代入的角色的名称 `s3-write-go-bucket-role`，随后是任务的名称 `7e9894e088ad416eb5cab92afExample`。

```
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROA36C6WWEJ2YEXAMPLE:7e9894e088ad416eb5cab92afExample",
    "arn": "arn:aws:sts::123456789012:assumed-role/s3-write-go-bucket-
role/7e9894e088ad416eb5cab92afExample",
    ...
}
```

> **Note**
>
> 在 Amazon EC2 容器实例上运行代入角色的任务时，Amazon ECS 容器代理会将请求记录到该代理的审核日志中，该代理位于 `/var/log/ecs/audit.log.YYYY-MM-DD-HH` 格式的地址上。有关更多信息，请参阅任务 IAM 角色日志和记录跟踪记录的见解事件。

Amazon ECS 任务执行 IAM 角色

任务执行角色的 ，该角色授予 Amazon ECS 容器和 Fargate 代理代表您进行 AWS API 调用的权限。任务执行 IAM 角色是必需的，具体取决于任务的要求。您可以将多个任务执行角色用于与您的账户关联的服务不同目的。有关您的应用程序运行所需的 IAM 权限，请参阅 Amazon ECS 任务 IAM 角色。

以下是任务执行 IAM 角色的常见使用案例：

- 您的任务托管在 AWS Fargate 或外部实例上且：
  - 从 Amazon ECR 私有存储库中提取容器映像。
  - 在与运行任务的账户不同的账户中，从 Amazon ECR 私有存储库中提取容器映像。
  - 使用 `awslogs` 日志驱动程序将容器日志发送到 CloudWatch Logs。有关更多信息，请参阅 将 Amazon ECS 日志发送到 CloudWatch 。
- 您的任务托管在 AWS Fargate 或 Amazon EC2 实例上且：
  - 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权限。
  - 使用运行时监控。
  - 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 Secrets Manager 或 Systems Manager 权限。

> **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 提供了名为 AmazonECSTaskExecutionRolePolicy 的托管式策略，该策略包含上述常见应用场景所需的权限。有关更多信息，请参阅《AWS 托管式策略参考指南》中的 AmazonECSTaskExecutionRolePolicy。对于特殊应用场景，可能需要向任务执行角色添加内联策略。

Amazon ECS 控制台将会创建一个任务执行角色。您可以手动为任务附加该托管式 IAM 策略，以便 Amazon ECS 能够在引入未来功能和增强功能时添加相关权限。您可以使用 IAM 控制台搜索来搜索

ecsTaskExecutionRole，并查看您的账户是否已有该任务执行角色。有关更多信息，请参阅《IAM 用户指南》中的 IAM 控制台搜索。

如果以经过身份验证的用户身份拉取映像，则不太可能受到 Docker Hub 拉取速率限制变化的影响。有关更多信息，请参阅容器实例的私有注册表身份验证。

通过使用 Amazon ECR 和 Amazon ECR Public，您可以避开 Docker 施加的限制。如果您从 Amazon ECR 提取映像，这还有助于缩短网络提取时间，并减少流量离开您的 VPC 时的数据传输变化。

使用 Fargate 时，您必须使用 repositoryCredentials 向私有映像注册表进行身份验证。无法为 Fargate 上托管的任务设置 Amazon ECS 容器代理环境变量 ECS_ENGINE_AUTH_TYPE 或 ECS_ENGINE_AUTH_DATA 或修改 ecs.config 文件。有关更多信息，请参阅任务的私有注册表身份验证。

创建任务执行 角色

如果您的账户尚未具有任务执行角色，请使用以下步骤创建角色。

AWS Management Console

为 Elastic Container Service 创建服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，选择 Elastic Container Service，然后选择 Elastic Container Service 任务使用案例。
5. 选择下一步。
6. 在添加权限部分中，搜索 AmazonECSTaskExecutionRolePolicy，然后选择该策略。
7. 选择下一步。
8. 对于角色名称，输入 ecsTaskExecutionRole。
9. 检查该角色，然后选择创建角色。

AWS CLI

将所有####替换为您自己的信息。

1. 创建一个名为 ecs-tasks-trust-policy.json 的文件，其中包含要用于 IAM 角色的信任策略。该文件应包含以下内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用上一步中创建的信任策略创建命名为 ecsTaskExecutionRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskExecutionRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 将 AWS 托管 AmazonECSTaskExecutionRolePolicy 策略 附加到 ecsTaskExecutionRole 角色。

```
aws iam attach-role-policy \
      --role-name ecsTaskExecutionRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

创建该角色后，为该角色添加以下功能的附加权限。

功能	其他权限
使用 Secrets Manager 凭证访问您的容器映像专用存储库	私有注册表身份验证权限
使用 Systems Manager 或 Secrets Manager 传递敏感数据	Secrets Manager 或 Systems Manager 权限

功能	其他权限
让 Fargate 任务通过接口端点拉取 Amazon ECR 映像	Fargate 任务通过接口端点拉取 Amazon ECR 映像的权限
在 Amazon S3 存储桶中托管配置文件	Amazon S3 存储桶文件存储权限

私有注册表身份验证权限

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添加到任务执行角色。有关更多信息，请参阅添加和删除 IAM policy。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` - 仅当密钥使用自定义 KMS 密钥而不是原定设置密钥时才需要。您的自定义密钥的 Amazon 资源名称（ARN）必须添加为资源。

下面是添加所需权限的示例内联策略。

```
{
    "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"
            ]
        }
    ]
}
```

Secrets Manager 或 Systems Manager 权限

允许容器代理拉取所需 AWS Systems Manager 或 Secrets Manager 资源的权限。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。

使用 Secrets Manager

要提供对您创建的 Secrets Manager 密钥的访问权限，请将以下权限手动添加到任务执行角色。有关如何管理权限的信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

- `secretsmanager:GetSecretValue` – 在引用 Secrets Manager 密钥时是必需的。添加从 Secrets Manager 中检索密钥的权限。

以下示例策略添加了所需的权限。

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

使用 Systems Manager

> **Important**
>
> 对于使用 EC2 启动类型的任务，必须使用 ECS 代理配置变量 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true` 才能使用此功能。您可以在创建容器实例的过程中将其添加到 `./etc/ecs/ecs.config` 文件中，也可以将其添加到现有实例中，然后重新启动 ECS 代理。有关更多信息，请参阅 Amazon ECS 容器代理配置。

要提供对您创建的 Systems Manager Parameter Store 参数的访问权限，请将以下权限作为策略手动添加到任务执行角色。有关如何管理权限的信息，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

- `ssm:GetParameters` — 当您在任务定义中引用 Systems Manager Parameter Store 参数时是必需的。添加检索 Systems Manager 参数的权限。
- `secretsmanager:GetSecretValue` — 当您直接引用 Secrets Manager 密钥或者您的 System Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必需的。添加从 Secrets Manager 中检索密钥的权限。
- `kms:Decrypt` — 仅当您的密钥使用客户托管键而不是默认键时才需要。您的自定义密钥的 ARN 应添加为资源。添加解密客户托管密钥的权限。

以下示例策略添加了所需的权限：

```
{
  "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"
      ]
    }
  ]
}
```

Fargate 任务通过接口端点拉取 Amazon ECR 映像的权限

在 Amazon ECR 配置为使用接口 VPC 端点的情况下，当启动使用 Fargate 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 端点的访问。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 端点的访问。有关更多信息，请参阅 AWS Global Condition Context Keys。

- `aws:SourceVpc` - 限制对特定 VPC 的访问。
- `aws:SourceVpce` - 限制对特定 VPC 端点的访问。

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 S3 存储桶文件存储权限

指定某个在 Amazon S3 中托管的配置文件时，任务执行角色必须包含对该配置文件的 `s3:GetObject` 权限以及对该文件所在 Amazon S3 存储桶的 `s3:GetBucketLocation` 权限。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的在策略中指定权限。

以下示例策略添加了从 Amazon S3 中检索文件所需的权限。指定 Amazon S3 存储桶的名称和配置文件名称。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket"
      ]
    }
  ]
}
```

Amazon ECS 任务 IAM 角色

您的 Amazon ECS 任务可以具有与其关联的 IAM 角色。在 IAM 角色中被授予的权限由任务中运行的容器承担。此角色允许您的应用程序代码（在容器上）使用其他 AWS 服务。当您的应用程序访问其他 AWS 服务（例如 Amazon S3）时需要任务角色。有关 Amazon ECS 提取容器映像和运行任务所需的 IAM 权限，请参阅 Amazon ECS 任务执行 IAM 角色。

使用任务角色有以下好处：

- 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。
- 审核：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。

> **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 Amazon EC2 或它们运行所在的外部实例继承任何 IAM 权限。

创建任务 IAM 角色

在创建供任务使用的 IAM 策略时，该策略必须包括您希望任务中的容器承担的权限。您可以使用现有 AWS 托管策略，也可以从头开始创建满足特定需求的自定义策略。有关更多信息，请参阅《IAM 用户指南》中的创建 IAM 策略。

> **Important**
>
> 对于 Amazon ECS 任务（适用于所有启动类型），我们建议您为任务使用 IAM policy 和角色。这些凭证允许您的任务在不调用 `sts:AssumeRole` 的情况下发出 AWS API 请求，以担任已与任务关联的相同角色。如果您的任务需要能代入自己的角色，则必须创建明确允许该角色代入自己的信任策略。有关更多信息，请参阅《IAM 用户指南》中修改角色信任策略。

创建 IAM policy 后，您可以创建 IAM 角色，其中包括您在 Amazon ECS 任务定义中引用的策略。您可以在 IAM 控制台中使用 Elastic Container Service 任务使用案例创建该角色。然后，可以将您特定的 IAM 策略附加到该角色，其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。

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。

有关您所在区域的服务端点的信息，请参阅《Amazon Web Services 一般参考 指南》中的服务端点。

IAM 任务角色必须具有指定 `ecs-tasks.amazonaws.com` 服务的信任策略。`sts:AssumeRole` 权限允许您的任务承担与 Amazon EC2 实例使用的不同的 IAM 角色。这样，您的任务不会继承与

Amazon EC2 实例关联的角色。以下是信任策略的示例。替换区域标识符并指定启动任务时使用的 AWS 账号。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

Important

创建任务 IAM 角色时，建议您在信任关系或与角色关联的 IAM 策略中使用 `aws:SourceAccount` 或 `aws:SourceArn` 条件键进一步限制权限范围，以防止混淆代理安全问题。使用 `aws:SourceArn` 条件键指定当前不受支持的特定集群时，应使用通配符来指定所有集群。要了解更多关于混淆代理问题以及如何保护您的 AWS 账户的信息，请参阅《IAM 用户指南》中的混淆代理问题。

以下过程通过示例策略介绍如何创建策略，以从 Amazon S3 检索对象。将所有####替换为您自己的值。

AWS Management Console

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

AWS CLI

将所有####替换为您自己的值。

1. 使用以下内容创建名为 s3-policy.json 的文件。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

2. 使用以下命令通过 JSON 策略文档文件创建 IAM 策略。

```
aws iam create-policy \
      --policy-name taskRolePolicy \
      --policy-document file://s3-policy.json
```

以下过程介绍如何通过附加您创建的 IAM 策略来创建任务 IAM 角色。

AWS Management Console

为 Elastic Container Service 创建服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，选择 Elastic Container Service，然后选择 Elastic Container Service 任务使用案例。
5. 选择下一步。
6. 对于添加权限，搜索并选择您创建的策略。
7. 选择下一步。
8. 对于角色名称，请为您的角色输入一个名称。在此示例中，键入 `AmazonECSTaskS3BucketRole` 以给角色命名。
9. 检查该角色，然后选择创建角色。

AWS CLI

将所有####替换为您自己的值。

1. 创建一个名为 `ecs-tasks-trust-policy.json` 的文件，其中包含要用于 IAM 角色的信任策略。该文件应包含以下内容。替换区域标识符并指定启动任务时使用的 AWS 账号。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
```

```
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

2. 使用上一步中创建的信任策略创建命名为 ecsTaskRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 使用以下命令检索您创建的 IAM 策略的 ARN。将 *taskRolePolicy* 替换为您创建的策略的名称。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'
```

4. 将创建的 IAM 策略其附加到 ecsTaskRole 角色。将 policy-arn 替换为所创建策略的 ARN。

```
aws iam attach-role-policy \
      --role-name ecsTaskRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy
```

创建该角色后，为该角色添加以下功能的附加权限。

功能	其他权限
使用 ECS Exec	ECS Exec 权限
使用 EC2 实例（Windows 和 Linux）	Amazon EC2 实例附加配置
使用外部实例	外部实例附加配置
使用 Windows EC2 实例	Amazon EC2 Windows 实例附加配置

ECS Exec 权限

ECS Exec 功能需要一个任务 IAM 角色来授予容器在托管 SSM 代理（`execute-command` 代理）和 SSM 服务之间进行通信所需的权限。您应向任务 IAM 角色添加以下权限，并在任务定义中包含任务 IAM 角色。有关更多信息，请参阅 IAM 用户指南中的添加和删除 IAM 策略。

对于您的任务 IAM 角色使用以下策略来添加所需的 SSM 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

Amazon EC2 实例附加配置

建议您将容器实例角色中的权限限制在 `AmazonEC2ContainerServiceforEC2Role` 托管 IAM policy 中使用的最低权限列表的范围内。

Amazon EC2 需要至少为版本 `1.11.0` 的容器代理才能使用任务角色；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。如果您使

用经 Amazon ECS 优化的 AMI，您的实例需要至少为 1.11.0-1 版的 ecs-init 程序包。如果您的实例使用最新的经 Amazon ECS 优化的 AMI，则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 经 Amazon ECS 优化的 Linux AMI。

如果对容器实例使用的不是经 Amazon ECS 优化的 AMI，请将 --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以下代理配置变量（有关更多信息，请参阅 Amazon ECS 容器代理配置）：

ECS_ENABLE_TASK_IAM_ROLE=true

为具有 bridge 和 default 网络模式的容器使用任务的 IAM 角色。

ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true

为具有 host 网络模式的容器使用任务的 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（适用于非经 Amazon ECS 优化的 AMI）。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。

要防止使用 awsvpc 网络模式的任务运行的容器访问提供给 Amazon EC2 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。

通过在 Amazon EC2 实例上运行以下 iptables 命令，阻止使用 bridge 网络模式的任务运行的容器访问提供给 Amazon EC2 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。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。

- ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER-USER 1 --in-
  interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```
  ```

您必须将此 iptables 规则保存到 Amazon EC2 实例上，使其在重启后仍然可用。使用经 Amazon ECS 优化的 AMI，您可以使用以下命令。对于其他操作系统，请参阅该操作系统的相关文档。

```
sudo iptables-save | sudo tee /etc/sysconfig/iptables && sudo systemctl enable --now
 iptables
```

外部实例附加配置

外部实例需要至少为版本 `1.11.0` 的容器代理才能使用任务 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。如果您正在使用经 Amazon ECS 优化的 AMI，您的实例将需要至少为 `1.11.0-1` 版的 `ecs-init` 程序包。如果您的实例使用最新的经 Amazon ECS 优化的 AMI，则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 经 Amazon ECS 优化的 Linux AMI。

如果对容器实例使用的不是经 Amazon ECS 优化的 AMI，请将 `--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以下代理配置变量（有关更多信息，请参阅 Amazon ECS 容器代理配置）：

`ECS_ENABLE_TASK_IAM_ROLE=true`

为具有 `bridge` 和 `default` 网络模式的容器使用任务的 IAM 角色。

`ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true`

为具有 `host` 网络模式的容器使用任务的 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（适用于非经 Amazon ECS 优化的 AMI）。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。

Amazon EC2 Windows 实例附加配置

> **Important**
>
> 这仅适用于 EC2 上使用任务角色的 Windows 容器。

具有 Windows 功能的任务角色需要在 EC2 上进行附加配置。

- 在启动容器实例时，您必须在容器实例用户数据脚本中设置 `-EnableTaskIAMRole` 选项。`EnableTaskIAMRole` 为任务打开任务 IAM 角色功能。例如：

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
</powershell>
```

- 您必须使用Amazon ECS 容器引导脚本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 创建任务 IAM 角色。
- 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，如果在容器实例上配置任务的 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。您可以在负载均衡器上使用端口 80。通过这样做，可以将流量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 使用负载均衡分配 Amazon ECS 服务流量。
- 如果重新启动 Windows 实例，则必须删除代理接口并再次初始化 Amazon ECS 容器代理以备份凭证代理。

Amazon ECS 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。

> **Note**
>
> 在 Windows 上使用 `awsvpc` 网络模式时，不需要运行此脚本。

如果您运行包含 Powershell 的 Windows 容器，请使用以下脚本：

```
# Copyright 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 -PolicyStore ActiveStore # credentials API
New-NetRoute -DestinationPrefix 169.254.169.254/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # metadata API
```

如果您运行的 Windows 容器只有命令 shell，请使用以下脚本：

```
# Copyright 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.

for /f "tokens=1" %i  in ('netsh interface ipv4 show interfaces ^| findstr /x /r
 ".*vEthernet.*"') do set interface=%i
for /f "tokens=3" %i  in ('netsh interface ipv4 show addresses %interface% ^| findstr /
x /r ".*Default.Gateway.*"') do set gateway=%i
netsh interface ipv4 add route prefix=169.254.170.2/32 interface="%interface%"
 nexthop="%gateway%" store=active # credentials API
```

```
netsh interface ipv4 add route prefix=169.254.169.254/32 interface="%interface%"
 nexthop="%gateway%" store=active # metadata API
```

Amazon ECS 容器实例 IAM 角色

Amazon ECS 容器实例（包括 Amazon EC2 和外部实例）运行 Amazon ECS 容器代理，并需要服务的 IAM 角色，以便了解属于您的代理。在启动容器实例并将其注册到集群之前，必须为容器实例创建 IAM 角色以供使用。该角色在您用于登录控制台或运行 AWS CLI 命令的账户中创建。

> **Important**
>
> 如果要将外部实例注册到集群，则您使用的 IAM 角色也需要 Systems Manager 权限。有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色。

Amazon ECS 提供 `AmazonEC2ContainerServiceforEC2Role` 托管 IAM policy，该策略包含使用完整 Amazon ECS 功能集所需的权限。此托管策略可以附加到 IAM 角色并与您的容器实例相关联。或者，您可以在创建要使用的自定义策略时使用托管策略作为指导。容器实例角色为 Amazon ECS 容器代理和 Docker 守护进程提供代表您调用 AWS API 所需的权限。有关托管策略的更多信息，请参阅 AmazonEC2ContainerServiceforEC2Role。

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。使用此功能时，建议您创建两个容器实例角色。为一个角色启用 `awsvpcTrunking` 账户设置，并将该角色用于需要 ENI 中继的任务。有关 `awsvpcTrunking` 账户设置的信息，请参阅通过账户设置访问 Amazon ECS 功能。

创建容器实例角色

> **Important**
>
> 如果要将外部实例注册到集群，请参阅 Amazon ECS Anywhere IAM 角色。

您可以手动为容器实例创建角色和附加托管 IAM policy，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。如果需要，可使用以下过程附加托管 IAM 策略。

AWS Management Console

为 Elastic Container Service 创建服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，选择 Elastic Container Service，然后选择 Elastic Container Service 的 EC2 角色使用案例。
5. 选择下一步。
6. 在权限策略部分，确认是否选择了 AmazonEC2ContainerServiceforEC2Role 策略。

> **Important**
>
> AmazonEC2ContainerServiceforEC2Role 托管策略应附加到容器实例 IAM 角色，否则使用 AWS Management Console 创建集群时将收到错误。

7. 选择下一步。
8. 对于角色名称，输入 ecsInstanceRole
9. 检查该角色，然后选择创建角色。

AWS CLI

将所有####替换为您自己的值。

1. 创建以下内容的名为 `instance-role-trust-policy.json` 的文件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用以下命令通过信任策略文档创建实例 IAM 角色。

```
aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://instance-role-trust-policy.json
```

3. 使用 create-instance-profile 命令创建名为 ecsInstanceRole-profile 的实例配置文件。

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole-profile
```

响应示例

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJTLBPJLEGREXAMPLE",
        "Roles": [],
        "CreateDate": "2022-04-12T23:53:34.093Z",
        "InstanceProfileName": "ecsInstanceRole-profile",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole-
profile"
    }
}
```

4. 将 *ecsInstanceRole* 角色添加到 *ecsInstanceRole-profile* 实例配置文件。

```
aws iam add-role-to-instance-profile \
    --instance-profile-name ecsInstanceRole-profile \
    --role-name ecsInstanceRole
```

5. 使用以下命令以将 AmazonEC2ContainerServiceRoleForEC2Role 托管策略附加到角色。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

创建该角色后，为该角色添加以下功能的附加权限。

功能	其他权限
Amazon ECR 具有容器映像	Amazon ECR 权限
使用 CloudWatch Logs 监控容器实例	监控容器实例的权限
在 Amazon S3 存储桶中托管配置文件	Amazon S3 只读访问权限

Amazon ECR 权限

用于容器实例的 Amazon ECS 容器实例角色必须具有以下有关 Amazon ECR 的 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 开发人员指南中的 Amazon ECS 容器实例 IAM 角色。

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 AWS CLI Amazon EC2 用户数据安装 ，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 Amazon ECS 容器实例配置存储在 Amazon S3 中。

可以使用以下 AWS CLI 命令允许 Amazon S3 对容器实例角色的只读访问。将 *ecsInstanceRole* 替换为您创建的角色的名称。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws::iam::aws:policy/AmazonS3ReadOnlyAccess
```

您也可以使用 IAM 控制台将 Amazon S3 只读访问权限（AmazonS3ReadOnlyAccess）添加到您的角色。有关更多信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色信任策略（控制台）。

监控容器实例的权限

您必须先创建 IAM policy 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,。

AWS Management Console

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
```

```
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

创建策略后，将策略附加到容器实例角色。有关如何将策略附加到角色的信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色权限策略（控制台）。

AWS CLI

1. 使用以下内容创建名为 `instance-cw-logs.json` 的文件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

2. 使用以下命令通过 JSON 策略文档文件创建 IAM 策略。

```
aws iam create-policy \
      --policy-name cwlogspolicy \
      --policy-document file://instance-cw-logs.json
```

3. 使用以下命令检索您创建的 IAM 策略的 ARN。将 *cwlogspolicy* 替换为您创建的策略的名称。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogsppolicy`].Arn'
```

4. 使用以下命令通过策略 ARN 将策略附加到容器实例 IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cwlogspolicy
```

Amazon ECS Anywhere IAM 角色

将本地部署服务器或虚拟机（VM）注册到集群时，服务器或虚拟机需要 IAM 角色才能与 AWS API 通信。您只需为每个创建此 IAM 角色一次 AWS 账户。但是，此 IAM 角色必须与您注册到群集的每个服务器或虚拟机关联。此角色是 ECSAnywhereRole。您可以手动创建此角色。Amazon ECS 可以在您在 AWS Management Console 中注册外部实例时代表您创建角色。您可以使用 IAM 控制台搜索来搜索 ecsAnywhereRole 并查看您的账户是否已有该角色。有关更多信息，请参阅《IAM 用户指南》中的 IAM 控制台搜索。

AWS 提供两个托管 IAM policy，可在创建 ECS Anywhere IAM 角色、AmazonSSMManagedInstanceCore 和 AmazonEC2ContainerServiceforEC2Role 策略时使用。AmazonEC2ContainerServiceforEC2Role 策略包含的权限可能会提供超出您需要的访问权限。因此，根据您的特定用例，我们建议您创建一个自定义策略，仅添加该策略中所需的权限。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。

任务执行 IAM角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。使用任务执行 IAM 角色时，必须在任务定义中指定该角色。有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色。

如果满足以下条件之一，则需要任务执行角色：

- 您正在使用 awslogs 日志驱动程序将容器日志发送到 CloudWatch Logs。

- 您的任务定义指定托管在 Amazon ECR 专用存储库中的容器映像。但是，如果与外部实例关联的 ECSAnywhereRole 角色还包含从 Amazon ECR 拉取映像所需的权限，那么任务执行角色不需要包括这些权限。

创建 Amazon ECS Anywhere 角色

将所有####替换为您自己的信息。

1. 利用以下信任策略创建名为 ssm-trust-policy.json 的本地文件。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": [
      "ssm.amazonaws.com"
    ]},
    "Action": "sts:AssumeRole"
  }
}
```

2. 使用以下 AWS CLI 命令创建角色并附加信任策略。

```
aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
 file://ssm-trust-policy.json
```

3. 使用以下命令附加 AWS 托管策略。

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
```

您还可以使用 IAM 自定义信任策略工作流来创建该角色。有关更多信息，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

Amazon ECS 基础设施 IAM 角色

Amazon ECS 基础设施 IAM 角色允许 Amazon ECS 代表您管理集群中的基础设施资源，并在以下情况使用：

- 您想将 Amazon EBS 卷附加到您的 Fargate 或 EC2 启动类型 Amazon ECS 任务。基础设施角色允许 Amazon ECS 为您的任务管理 Amazon EBS 卷。
- 您想使用传输层安全性协议（TLS）加密您的 Amazon ECS Service Connect 服务之间的流量。

当 Amazon ECS 担任此角色代表您采取行动时，这些事件将在 AWS CloudTrail 中可见。如果 Amazon ECS 使用该角色来管理附加到任务的 Amazon EBS 卷，则 CloudTrail 日志 `roleSessionNamefolefole` 将为 `ECSTaskVolumesForEBS`。如果该角色用于加密您的 Amazon ECS Service Connect 服务之间的流量，则 CloudTrail 日志 `roleSessionName` 将为 `ECSServiceConnectForTLS`。您可以使用此名称通过筛选用户名在 CloudTrail 控制台中搜索事件。

Amazon ECS 提供了若干托管式策略，其中包含卷挂载和 TLS 所需的权限。有关更多信息，请参阅《AWS 托管式策略参考指南》中的 AmazonECSInfrastructureRolePolicyForVolumes 和 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity。

创建 Amazon ECS 基础设施角色

将所有####替换为您自己的信息。

1. 创建一个名为 `ecs-infrastructure-trust-policy.json` 的文件，其中包含要用于 IAM 角色的信任策略。该文件应包含以下内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToECSForInfrastructureManagement",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用您在上一步中创建的信任策略，并使用以下 AWS CLI 命令创建一个名为 `ecsInfrastructureRole` 的角色。

```
aws iam create-role \
      --role-name ecsInfrastructureRole \
```

```
      --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. 根据您的使用案例，将 AWS 托管的 AmazonECSInfrastructureRolePolicyForVolumes 或 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 策略附加到 ecsInfrastructureRole 角色。

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForVolumes
```

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity
```

您也可以使用 IAM 控制台的自定义信任策略工作流程来创建该角色。有关更多信息，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

Important

如果 Amazon ECS 使用 ECS 基础设施角色来管理附加到您的任务的 Amazon EBS 卷，则请在停止使用 Amazon EBS 卷的任务之前确保满足以下条件。

- 该角色未被删除。
- 该角色的信任策略不会被修改为删除 Amazon ECS 访问权限（ecs.amazonaws.com）。
- 托管策略 AmazonECSInfrastructureRolePolicyForVolumes 未被删除。如果您必须修改角色的权限，则请至少保留 ec2:DetachVolume、ec2:DeleteVolume 和 ec2:DescribeVolumes 用于卷删除。

在停止带有附加 Amazon EBS 卷的任务之前删除或修改角色将导致任务陷入 DEPROVISIONING 状态，并且关联的 Amazon EBS 卷无法删除。Amazon ECS 将定期自动重试以停止任务并删除该卷，直到恢复必要的权限。您可以使用 DescribeTasks API 查看任务的卷附加状态和关联的状态原因。

创建该文件后，您必须向用户授予将该角色传递给 Amazon ECS 的权限。

授予将基础设施角色传递给 Amazon ECS 的权限

要使用 ECS 基础设施 IAM 角色，您必须授予用户将该角色传递给 Amazon ECS 的权限。将以下 `iam:PassRole` 权限附加到您的用户。将 *`ecsInfrastructureRole`* 替换为您创建的基础设施角色的名称。

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

有关 `iam:Passrole` 和更新用户权限的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的向用户授予权限以将角色传递给 AWS 服务和更改 IAM 用户的权限。

Amazon ECS CodeDeploy IAM 角色

在将 CodeDeploy 蓝色/绿色部署类型用于 Amazon ECS 之前，CodeDeploy 服务需要获得代表您更新 Amazon ECS 服务的权限。这些权限是由 CodeDeploy IAM 角色 (`ecsCodeDeployRole`) 提供的。

> **Note**
>
> 用户还需要使用 CodeDeploy 的权限；所需的 IAM 权限 中介绍了这些权限。

提供了两个托管策略。有关更多信息，请参阅《AWS 托管式策略参考指南》中的以下章节之一：

- AWSCodeDeployRoleForECS – 向 Codeploy 授予使用相关操作更新任何资源的权限。
- AWSCodeDeployRoleForECSLimited – 向 Codeploy 授予更多有限权限。

创建 CodeDeploy 角色

您可以使用以下过程为 Amazon ECS 创建 CodeDeploy 角色

AWS Management Console

创建用于 CodeDeploy 的服务角色（IAM 控制台）

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。
4. 对于服务或使用案例，请选择 CodeDeploy，然后选择 CodeDeploy - ECS 使用案例。
5. 选择下一步。
6. 在附加权限策略部分中，确保选择了 AWSCodeDeployRoleForECS 策略。
7. 选择下一步。
8. 对于在角色名称，输入 ecsCodeDeployRole。
9. 检查该角色，然后选择创建角色。

AWS CLI

将所有####替换为您自己的信息。

1. 创建一个名为 `codedeploy-trust-policy.json` 的文件，其中包含要用于 CodeDeploy IAM 角色的信任策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": ["codedeploy.amazonaws.com"]
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

2. 使用上一步中创建的信任策略创建命名为 ecsCodedeployRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsCodedeployRole \
      --assume-role-policy-document file://codedeploy-trust-policy.json
```

3. 将 AWSCodeDeployRoleForECS 或 AWSCodeDeployRoleForECSLimited 托管策略附加到 ecsTaskRole 角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECS
```

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws::iam::aws:policy/AWSCodeDeployRoleForECSLimited
```

当服务中的任务需要某个任务执行角色时，必须将每个任务执行角色或任务角色覆盖的 iam:PassRole 权限作为策略添加到 Codeploy 角色。

任务执行角色权限

当服务中的任务需要某个任务执行角色时，必须将每个任务执行角色或任务角色覆盖的 iam:PassRole 权限作为策略添加到 Codeploy 角色。有关更多信息，请参阅Amazon ECS 任务执行 IAM 角色 和Amazon ECS 任务 IAM 角色。然后，将该策略附加到该 CodeDeploy 角色

创建策略

AWS Management Console

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。

4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

创建策略后，将策略附加到 CodeDeploy 角色。有关如何将策略附加到角色的信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色权限策略（控制台）。

AWS CLI

将所有####替换为您自己的信息。

1. 使用以下内容创建名为 `blue-green-iam-passrole.json` 的文件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

2. 使用以下命令通过 JSON 策略文档文件创建 IAM 策略。

```
aws iam create-policy \
      --policy-name cdTaskExecutionPolicy \
      --policy-document file://blue-green-iam-passrole.json
```

3. 使用以下命令检索您创建的 IAM 策略的 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. 使用以下命令将策略附加到 CodeDeploy IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cdTaskExecutionPolicy
```

Amazon ECS EventBridge IAM 角色

在您能够将 Amazon ECS 计划任务与 EventBridge 规则和目标配合使用之前，EventBridge 服务需要代表您运行 Amazon ECS 任务的权限。这些权限由 EventBridge IAM 角色（`ecsEventsRole`）提供。

`AmazonEC2ContainerServiceEventsRole` 策略如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
            }
        }
    ]
}
```

如果计划任务需要使用任务执行角色、任务角色或任务角色覆盖，则必须将每个任务执行角色、任务角色或任务角色覆盖的 `iam:PassRole` 权限添加到 EventBridge IAM 角色。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。

> **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>"]
        }
    ]
}
```

配置计划任务时，您可以选择让 AWS Management Console 创建 EventBridge 角色。有关更多信息，请参阅 使用 Amazon EventBridge 调度器计划 Amazon ECS 任务 。

创建 EventBridge 角色

将所有####替换为您自己的信息。

1. 创建一个名为 eventbridge-trust-policy.json 的文件，其中包含要用于 IAM 角色的信任策略。该文件应包含以下内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用以下命令创建一个 IAM 角色，该角色使用您在上一步中创建的信任策略命名为 ecsEventsRole。

```
aws iam create-role \
      --role-name ecsEventsRole \
      --assume-role-policy-document file://eventbridge-policy.json
```

3. 使用以下命令将 AWS 托管 AmazonEC2ContainerServiceEventsRole 附加到 ecsEventsRole 角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceEventsRole
```

您也可以使用 IAM 控制台的自定义信任策略工作流程（https://console.aws.amazon.com/iam/）来创建该角色。有关更多信息，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

将策略附加到 **ecsEventsRole** 角色

您可以使用以下过程将任务执行角色的权限添加到 EventBridge IAM 角色。

AWS Management Console

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

创建策略后，将策略附加到 EventBridge 角色。有关如何将策略附加到角色的信息，请参阅《AWS Identity and Access Management 用户指南》中的修改角色权限策略（控制台）。

AWS CLI

将所有####替换为您自己的信息。

1. 使用以下内容创建名为 ev-iam-passrole.json 的文件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

2. 使用以下 AWS CLI 命令通过 JSON 策略文档文件创建 IAM 策略。

```
aws iam create-policy \
      --policy-name eventsTaskExecutionPolicy \
      --policy-document file://ev-iam-passrole.json
```

3. 使用以下命令检索您创建的 IAM 策略的 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'
```

4. 使用以下命令通过策略 ARN 将策略附加到 EventBridge IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/eventsTaskExecutionPolicy
```

Amazon ECS 控制台所需的权限

遵循授予最低权限的最佳实践，您可以使用 `AmazonECS_FullAccess` 托管策略作为创建您自己的自定义策略的模板。这样，您就可以根据您的特定要求取消或添加托管策略的权限。有关更多信息，请参阅 权限详细信息。

在以下情况下，Amazon ECS 控制台由 AWS CloudFormation 提供支持，并且需要额外的 IAM 权限：

- 创建集群
- 创建服务
- 创建容量提供程序

您可以为其他权限创建策略，然后将其附加到用于访问控制台的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的创建 IAM 策略。

创建集群所需的权限

在控制台中创建集群时，您需要额外的权限来授予您管理 AWS CloudFormation 堆栈的权限。

需要以下额外权限：

- `cloudformation`— 允许委托人创建和管理 AWS CloudFormation 堆栈。这是使用 AWS Management Console 创建 Amazon ECS 群集和后续管理这些群集时所必需的。

以下策略包含所需的 AWS CloudFormation 权限，并将操作限制为在 Amazon ECS 控制台中创建的资源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
            ],
           "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
           ]
```

```
      }
   ]
}
```

如果您尚未创建 Amazon ECS 容器实例角色（ecsInstanceRole），并且正在创建使用 Amazon EC2 实例的集群，则控制台将代表您创建该角色。

此外，如果您使用自动扩缩组，则需要额外的权限，以便控制台在使用集群自动扩缩功能时可以向自动扩缩组添加标签。

需要以下额外权限：

- autoscaling – 允许控制台标记 Amazon EC2 Auto Scaling 组。使用集群 Auto Scaling 功能时，在管理 Amazon EC2 Auto Scaling 组时需要此项。该标签是 ECS 管理的标签，控制台自动将其添加到组中，以表示它是在控制台中创建的。
- iam— 允许委托人列出 IAM 角色及其附加的策略。委托人还可以列出 Amazon EC2 实例可用的实例配置文件。

以下策略包含所需的 IAM 权限，并将操作限制到 ecsInstanceRole 角色。

自动扩缩权限不受限制。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateOrUpdateTags",
            "Resource": "*"
        }
```

```
    ]
}
```

创建容量提供程序所需的权限

在控制台中创建服务时，您需要额外的权限来授予您管理 AWS CloudFormation 堆栈的权限。需要以下额外权限：

- `cloudformation`— 允许委托人创建和管理 AWS CloudFormation 堆栈。这是使用 AWS Management Console 创建 Amazon ECS 容量提供程序以及随后管理这些容量提供程序时所必需的。

以下策略包含所需的权限，并将操作限制为在 Amazon ECS 控制台中创建的资源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-CapacityProvider-*"
            ]
      }
   ]
}
```

创建服务所需的权限

在控制台中创建服务时，您需要额外的权限来授予您管理 AWS CloudFormation 堆栈的权限。需要以下额外权限：

- `cloudformation`— 允许委托人创建和管理 AWS CloudFormation 堆栈。这是使用 AWS Management Console 创建 Amazon ECS 服务和后续管理这些服务时所必需的。

以下策略包含所需的权限，并将操作限制为在 Amazon ECS 控制台中创建的资源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
            ]
      }
   ]
}
```

创建 IAM 角色的权限

以下操作需要额外的权限才能完成操作：

- 注册外部实例 - 有关更多信息，请参阅 Amazon ECS Anywhere IAM 角色
- 注册任务定义 - 有关更多信息，请参阅 Amazon ECS 任务执行 IAM 角色
- 创建用于计划任务的 EventBridge 规则 - 有关更多信息，请参阅 Amazon ECS EventBridge IAM 角色

您可以通过先在 IAM 中创建角色来添加这些权限，然后再在 Amazon ECS 控制台中使用这些权限。如果您没有创建角色，Amazon ECS 控制台将代表您创建角色。

将外部实例注册到集群所需的权限

在向集群注册外部实例并想要创建新的外部实例（`escExternalInstanceRole`）角色时，您需要额外的权限。

需要以下额外权限：

- `iam` – 允许主体创建并列出 IAM 角色及其附加的策略。
- `ssm` – 允许主体向 Systems Manager 注册外部实例。

> **Note**
>
> 要选择现有的 escExternalInstanceRole，您必须拥有 iam:GetRole 和 iam:PassRole 权限。

以下策略包含所需的权限，并将操作限制到 escExternalInstanceRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole","ssm:CreateActivation"],
            "Resource": "arn:aws:iam::*:role/escExternalInstanceRole"
        }
    ]
}
```

注册任务定义所需的权限

注册任务定义并想要创建新的任务执行（ecsTaskExecutionRole）角色时，您需要额外的权限。

需要以下额外权限：

- iam – 允许主体创建并列出 IAM 角色及其附加的策略。

> **Note**
>
> 要选择现有的 `ecsTaskExecutionRole`，您必须拥有 `iam:GetRole` 权限。

以下策略包含所需的权限，并将操作限制到 `ecsTaskExecutionRole` 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
        }
    ]
}
```

为计划任务创建 EventBridge 规则所需的权限

在计划任务并想要创建新的 CloudWatch Events 角色（`ecsEventsRole`）角色时，您需要额外的权限。

需要以下额外权限：

- `iam`– 允许主体创建和列出 IAM 角色及其附加策略，并允许 Amazon ECS 将角色传递给其他服务以代入该角色。

> **Note**
>
> 要选择现有的 `ecsEventsRole`，您必须拥有 `iam:GetRole` 和 `iam:PassRole` 权限。

以下策略包含所需的权限，并将操作限制到 `ecsEventsRole` 角色。

```
{
```

```
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole",
              "iam: PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsEventsRole"
        }
    ]
}
```

Amazon ECS 服务自动扩缩所需的 IAM 权限

通过 Amazon ECS、CloudWatch 和 Application Auto Scaling API 的组合，服务自动扩展成为可能。使用 Amazon ECS 创建和更新服务，使用 CloudWatch 创建警报，使用 Application Auto Scaling 创建扩展策略。

除了用于创建和更新服务的标准 IAM 权限外，与 Service Auto Scaling 设置进行交互还需要以下权限，如以下示例策略所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ecs:DescribeServices",
                "ecs:UpdateService",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
```

```
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": ["*"]
        }
    ]
}
```

创建 Amazon ECS 服务示例 和 更新 Amazon ECS 服务示例 IAM policy 示例显示在 AWS Management Console 中使用 Service Auto Scaling 所需的权限。

Application Auto Scaling 服务还需要描述 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。`sns:` 权限用于在超过阈值时 CloudWatch 向 Amazon SNS 主题发送的通知。如果您为 Amazon ECS 服务使用弹性伸缩功能，它将创建一个名为 `AWSServiceRoleForApplicationAutoScaling_ECSService` 的服务相关角色。此服务相关角色授予 Application Auto Scaling 权限，以描述策略警报、监控服务的当前运行的任务数以及修改服务的所需计数。Application Auto Scaling 的原托管 Amazon ECS角色为 `ecsAutoscaleRole`，但今后已不再需要。此服务相关角色是 Application Auto Scaling 的默认角色。有关更多信息，请参阅《Application Auto Scaling 用户指南》中的 Application Auto Scaling 服务相关角色。

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 `ecs:StartTelemetrySession` 权限。有关更多信息，请参阅 注意事项。

在创建时授予标记资源的权限

以下标签创建 Amazon ECS API 操作允许您在创建资源时指定标签。如果在资源创建操作中指定了标签，则 AWS 会执行额外的授权，以验证是否分配了正确的权限以创建标签。

- `CreateCapacityProvider`
- `CreateCluster`
- `CreateService`
- `CreateTaskSet`
- `RegisterContainerInstance`
- `RegisterTaskDefinition`
- `RunTask`
- `StartTask`

您可以使用资源标签来实现基于属性的控制 (ABAC)。有关更多信息，请参阅the section called “使用资源标签控制对 Amazon ECS 资源的访问” 和标记资源。

为允许在创建时添加标签，请创建或修改策略，以包含使用创建该资源的操作（如 `ecs:CreateCluster` 或 `ecs:RunTask` 和 `ecs:TagResource`）的权限。

以下示例演示了一个策略，该策略使用户能够创建集群和在集群创建过程中添加标签。用户无权标记任何现有资源 (他们无法直接调用 `ecs:TagResource` 操作)。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:CreateCluster"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
                  "ecs:CreateAction": [
                      "CreateCluster",
                      "CreateCapacityProvider",
                      "CreateService",
                      "CreateTaskSet",
                      "RegisterContainerInstance",
                      "RegisterTaskDefinition",
                      "RunTask",
                      "StartTask"
                  ]
          }
       }
    }
  ]
}
```

仅当用户在资源创建操作中应用了标签时，系统才会评估 ecs:TagResource 操作。因此，如果未在此请求中指定任何标签，则拥有创建资源权限 (假定没有标记条件) 的用户无需具备使用 ecs:TagResource 操作的权限。但是，如果用户不具备使用 ecs:TagResource 操作的权限而又试图创建带标签的资源，则请求将失败。

Amazon ECS 控制对特定标签的访问

您可以在 IAM policy 的 Condition 元素中使用其他条件来控制可应用到资源的标签键和值。

以下条件键可用于上一节中的示例：

- aws:RequestTag：指示请求中必须存在特定的标签键或标签键和值。也可在此请求中指定其他标签。
  - 与 StringEquals 条件运算符配合使用，以强制实施特定的标签键和值组合，例如强制实施标签 cost-center=cc123：

    ```
    "StringEquals": { "aws:RequestTag/cost-center": "cc123" }
    ```

  - 与 StringLike 条件运算符配合使用，以在请求中强制实施特定的标签键；如强制实施标签键 purpose：

    ```
    "StringLike": { "aws:RequestTag/purpose": "*" }
    ```

- aws:TagKeys：强制实施在请求中使用的标签键。
  - 与 ForAllValues 修饰符配合使用，以只强制实施请求中提供的特定标签键 (如果在请求中指定了标签，则只允许特定的标签键；不允许任何其他标签)。例如，允许标签键 environment 或 cost-center：

    ```
    "ForAllValues:StringEquals": { "aws:TagKeys": ["environment","cost-center"] }
    ```

  - 与 ForAnyValue 修饰符配合使用，以强制请求中至少存在一个指定的标签键。例如，强制请求中至少存在标签键 environment 或 webserver 中的一个：

    ```
    "ForAnyValue:StringEquals": { "aws:TagKeys": ["environment","webserver"] }
    ```

上述条件键可应用于支持标记的资源创建操作，以及 ecs:TagResource 操作。要了解 Amazon ECS API 操作是否支持添加标签，请参阅 Amazon ECS 的操作、资源和条件键。

为强制用户指定标签，在创建资源时，您必须使用 aws:RequestTag 条件键或 aws:TagKeys 条件键，并在资源创建操作中使用修饰符 ForAnyValue。如果用户没有为资源创建操作指定标签，则不会对 ecs:TagResource 操作进行评估。

对于条件，条件键不区分大小写，条件值区分大小写。因此，要强制标签键区分大小写，请使用 aws:TagKeys 条件键，其中标签键指定为条件中的值。

有关多值条件的更多信息，请参阅IAM 用户指南中的创建测试多个键值的条件。

使用资源标签控制对 Amazon ECS 资源的访问

在创建向用户授予使用 Amazon ECS 资源的权限的 IAM policy 时，可以在该策略的 Condition 元素中包含标签信息，以根据标签控制访问权限。这称为基于属性的访问控制 (ABAC)。ABAC 可以让您更好地控制用户可以修改、使用或删除哪些 EC2 资源。有关更多信息，请参阅什么是适用于 AWS 的 ABAC？

例如，您可以创建一个策略，允许用户删除集群，但在集群具有 environment=production 标签时拒绝此操作。为此，您可以使用 aws:ResourceTag 条件键来基于附加到资源的标签允许或拒绝对资源的访问。

```
"StringEquals": { "aws:ResourceTag/environment": "production" }
```

要了解 Amazon ECS API 操作是否支持使用 aws:ResourceTag 条件键控制访问，请参阅 Amazon ECS 的操作、资源和条件键。请注意，Describe 操作不支持资源级权限，因此，您必须在不带条件的单独语句中指定它们。

有关示例 IAM policies，请参阅 Amazon ECS 示例策略 。

如果您基于标签允许或拒绝用户访问资源，则必须考虑显式拒绝用户对相同资源添加或删除这些标签的能力。否则，用户可能通过修改资源标签来绕过您的限制并获得资源访问权限。

Amazon ECS 示例策略

您可以使用 IAM policy 向用户授予在 Amazon ECS 控制台中查看和使用特定资源的权限。您可以使用上一部分中的策略；但是，这些策略设计用于使用 AWS CLI 或 AWS 开发工具包发出的请求。

示例：允许用户根据标签删除 Amazon ECS 集群

当标签的键/值对为“目的/测试”时，以下策略允许用户删除集群。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
       {
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ecs:region:account-id:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Testing"
        }
      }
    }
  ]
}
```

对 Amazon Elastic Container Service 标识和访问进行故障排除

可以使用以下信息，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
- 我无权执行 iam:PassRole
- 我希望允许我的 AWS 账户 以外的人访问我的 Amazon ECS 资源
- 其他故障排除资源

我无权在 Amazon ECS 中执行操作

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 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
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 ecs:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon ECS。

有些 AWS 服务 允许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此操作。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，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系 AWS 管理员。您的管理员是提供登录凭证的人。

我希望允许我的 AWS 账户 以外的人访问我的 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，可以担任角色。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 如何将 Amazon Elastic Container Service 与 IAM 结合使用。
- 要了解如何为您拥有的 AWS 账户中的资源提供访问权限，请参阅《IAM 用户指南》中的为您拥有的另一个 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅《IAM 用户指南》中的为第三方拥有的AWS 账户提供访问权限。
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）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存取之间的差别，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。

其他故障排除资源

以下页面提供了有关错误代码的信息：

- Amazon ECS 已停止任务错误消息
- 查看 Amazon ECS 服务事件消息

Amazon EC2 的 IAM 最佳实践

您可以使用 AWS Identity and Access Management（IAM）通过基于规则的策略来管理和控制对您的 AWS 服务和资源的访问权限，以实现身份验证和授权目的。更具体地说，通过此服务，您可以使用应用于用户、组或角色的策略来控制对 AWS 资源的访问权限。在这三者中，用户是可以访问您的资源的账户。IAM 角色是一组可由经过身份验证的身份承担的权限，该身份与 IAM 之外的特定身份无关。有关更多信息，请参阅访问管理的 Amazon ECS 概览：权限和策略。

遵循最低权限访问策略

创建限定范围的策略，以允许用户执行其规定的工作。例如，如果开发者需要定期停止某项任务，请创建一个仅允许该特定操作的策略。以下示例仅允许用户在具有特定 Amazon 资源名称（ARN）的集群上停止属于特定 `task_family` 的任务。在条件中引用 ARN 也是使用资源级权限的一个示例。您可以使用资源级权限来指定要向其应用操作的资源。

> **Note**
>
> 在策略中引用 ARN 时，请使用新的加长 ARN 格式。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon 资源名称（ARN）和 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:StopTask"
      ],
      "Condition": {
        "ArnEquals": {
          "ecs:cluster": "arn:aws:ecs:region:account_id:cluster/cluster_name"
```

```
        }
      },
      "Resource": [
        "arn:aws:ecs:region:account_id:task-definition/task_family:*"
      ]
    }
  ]
}
```

将集群资源作为管理边界

范围过于狭窄的策略可能会导致角色激增并增加管理开销。与其创建仅限于特定任务或服务的角色，不如创建限制于集群的角色并将该集群用作您的主要管理边界。

创建自动化管道将最终用户与 API 隔离开来

您可以通过创建管道来限制用户可以使用的操作，这些管道会自动打包应用程序并将其部署到 Amazon ECS 集群上。这会有效地将创建、更新和删除任务的工作委托给管道。有关更多信息，请参阅《AWS CodePipeline 用户指南》中的教程：使用 CodePipeline 进行 Amazon ECS 标准部署。

使用策略条件来增加一层安全性

当您需要增加安全层时，请在策略中添加一个条件。如果您正在执行特权操作或需要限制可对特定资源执行的一组操作，则这可能很有用。以下示例策略在删除集群时需要多因素授权。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      },
    "Resource": ["*"]
    }
  ]
}
```

应用于服务的标签会传播到属于该服务的所有任务。因此，您可以创建仅限于具有特定标签的 Amazon ECS 资源的角色。在以下策略中，IAM 主体启动和停止标签键为 `Department`、标签值为 `Accounting` 的所有任务。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Resource": "arn:aws:ecs:*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Department": "Accounting"}
            }
        }
    ]
}
```

定期审核对 API 的访问权限

用户可能会更改角色。在他们更改角色后，之前授予他们的权限可能不再适用。请务必审核谁有权访问 Amazon ECS API，以及该访问权限是否仍有保障。考虑将 IAM 与用户生命周期管理解决方案集成，该解决方案可在用户离开组织时自动撤消访问权限。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon ECS 安全组审核指南。

Amazon Elastic Container Service 中的日志记录和监控

监控是保持 Amazon Elastic Container Service 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。您应该从 AWS 解决方案的各个部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS 提供了多种工具来监控您的 Amazon ECS 资源并对潜在事件做出响应：

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。具体操作是：通知已发送到 Amazon Simple Notification Service（Amazon SNS）主题或 Amazon EC2 Auto Scaling 策略。CloudWatch 告警不调用操作，因为这些操作处于特定状态；状态必须改变并保持指定时间。有关更多信息，请参阅 使用 CloudWatch 监控 Amazon ECS 。

对于具有使用 Fargate 启动类型的任务的服务，您可以使用 CloudWatch 警报根据 CloudWatch 指标（如 CPU 和内存利用率）在服务中扩展或缩减任务。有关更多信息，请参阅 自动扩展 Amazon ECS 服务。

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）横向缩减和横向扩展容器实例。

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。有关更多信息，请参阅使用 awslogs 驱动程序。

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。

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 使用 EventBridge 自动响应 Amazon ECS 错误 以及 Amazon CloudWatch Events 用户指南中的什么是 Amazon CloudWatch Events？

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在 Amazon ECS 中所执行操作的记录。使用由 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 使用 AWS CloudTrail 记录 Amazon ECS API 调用。

AWS Trusted Advisor

Trusted Advisor 凝聚了从为数十万 AWS 客户提供服务中总结的最佳实践。Trusted Advisor 可检查您的 AWS 环境，然后在有可能节省开支、提高系统可用性和性能或弥补安全漏洞时为您提供建议。所有 AWS 客户均有权访问五个 Trusted Advisor 检查。使用“商业”和“企业”支持计划的客户可以查看所有 Trusted Advisor 检查。

有关更多信息，请参阅《AWS Support 用户指南》中的 AWS Trusted Advisor。

AWS Compute Optimizer

AWS Compute Optimizer 是一种服务，用于分析 AWS 资源的配置和利用率指标。它将报告您的资源是否处于最佳状态并生成优化建议，以降低成本并提高工作负载的性能。

有关更多信息，请参阅 针对 Amazon ECS 的 AWS Compute Optimizer 建议。

监控 Amazon ECS 的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。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。

Amazon Elastic Container Service 的合规性验证

要了解某个 AWS 服务 是否在特定合规性计划范围内，请参阅合规性计划范围内的 AWS 服务 ，然后选择您感兴趣的合规性计划。有关常规信息，请参阅AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告、。

您使用 AWS 服务 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署以安全性和合规性为重点的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计– 该白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

> **Note**
>
> 并非所有 AWS 服务都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS 客户合规指南：从合规角度了解责任共担模式。这些指南总结了保护 AWS 服务 的最佳实践，并将指南映射到跨多个框架的安全控制，包括美国国家标准与技术研究院（NIST）、支付卡行业安全标准委员会（PCI）和国际标准化组织（ISO）。
- AWS Config 开发人员指南中的使用规则评估资源 - 此 AWS Config 服务评测您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务 向您提供 AWS 中安全状态的全面视图。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty – 该 AWS 服务 通过监控您的环境中是否存在可疑和恶意活动，来检测您的 AWS 账户、工作负载、容器和数据面临的潜在威胁。GuardDuty 可以通过满足某些合规性框架规定的入侵检测要求，来协助您满足各种合规性要求，如 PCI DSS。

- AWS Audit Manager——此 AWS 服务 可帮助您持续审计您的 AWS 使用情况，以简化管理风险以及与相关法规和行业标准的合规性的方式。

Amazon ECS 的合规性和安全性最佳实践

您在使用 Amazon ECS 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。

AWS提供以下资源来帮助实现合规性：

- 安全性与合规性快速入门指南：这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书：此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- 合规性计划范围内的 AWS 服务：此列表包含在特定合规性计划范围内的 AWS 服务。有关更多信息，请参阅 AWS 合规性计划。

支付卡行业数据安全标准（PCI DSS）

在遵守 PCI DSS 时，了解环境中持卡人数据（CHD）的完整流程非常重要。CHD 流程决定了 PCI DSS 的适用性，定义了持卡人数据环境（CDE）的边界和组成部分，从而定义了 PCI DSS 评测的范围。准确确定 PCI DSS 范围是定义安全态势并最终成功评测的关键。客户必须有一个确定范围的程序，以确保其完整性并检测范围的变化或偏差。

容器化应用程序的临时性质在审计配置时增加了复杂性。因此，客户需要保持对所有容器配置参数的了解，以确保在容器生命周期的所有阶段都能满足合规性要求。

有关在 Amazon ECS 上实现 PCI DSS 合规性的更多信息，请参阅以下白皮书。

- 在 Amazon ECS 上进行设计以实现 PCI DSS 合规性
- 在 AWS 上设计 PCI DSS 范围和分段

HIPAA（美国健康保险流通与责任法案）

将 Amazon ECS 与处理受保护的健康信息（PHI）的工作负载一起使用无需额外配置。Amazon ECS 充当编排服务，用于编排容器在 Amazon EC2 上的启动。它不会使用正在编排的工作负载中的数据进行操作，也不会对这些数据进行操作。根据 HIPAA 法规和 AWS 商业伙伴增订合约，在使用 Amazon ECS 推出的容器访问时，PHI 应在传输和静态时进行加密。

每个 AWS 存储选项都提供了多种静态加密机制，例如 Amazon S3、Amazon EBS 和 AWS KMS。您可以部署叠加网络（例如 VNS3 或 Weave Net），以确保对容器之间传输的 PHI 进行完全加密，或者提供冗余的加密层。还应启用完整日志记录，并将所有容器日志定向到 Amazon CloudWatch。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

AWS Security Hub

使用 AWS Security Hub 监控 Amazon ECS 的使用情况，因为它与安全最佳实践有关。Security Hub 使用控件来评估资源配置和安全标准，以帮助您遵守各种合规框架。有关使用 Security Hub 评估 Amazon ECS 资源的更多信息，请参阅《AWS Security Hub 用户指南》中的 Amazon ECS 控件。

具有 Amazon ECS 运行时监控的 Amazon GuardDuty

Amazon GuardDuty 是一项威胁检测服务，可帮助保护您的账户、容器、工作负载和 AWS 环境中的数据。GuardDuty 使用机器学习（ML）模型以及异常和威胁检测功能，持续监控不同的日志源和运行时活动，以识别环境中的潜在安全风险和恶意活动并确定其优先级。

使用 GuardDuty 中的运行时监控来识别恶意或未经授权的行为。运行时监控通过持续监控 AWS 日志和联网活动来识别恶意或未经授权的行为，从而保护在 Fargate 和 EC2 上运行的工作负载。运行时监控使用轻量级、完全托管的 GuardDuty 安全代理分析主机中的行为，例如文件访问、进程执行和网络连接。它涉及的问题包括权限升级、使用公开的凭证，或与恶意 IP 地址、域通信，以及您的 Amazon EC2 实例和容器工作负载上存在恶意软件。有关更多信息，请参阅《GuardDuty User Guide》中的 GuardDuty Runtime Monitoring。

合规性建议

您应该尽早与企业内部的合规计划所有者接触，并使用 AWS 责任共担模型来确定合规控制所有权，以便成功实施相关的合规计划。

AWS Fargate 美国联邦信息处理标准（FIPS-140）

美国联邦信息处理标准（FIPS）。FIPS-140 是美国和加拿大政府标准，其中规定了对保护敏感信息的加密模块的安全要求。FIPS-140 定义了一组经过验证的加密函数，可用于加密传输中的数据和静态数据。

开启 FIPS-140 合规性后，您可以以符合 FIPS-140 的方式在 Fargate 上运行工作负载。有关 FIPS-140 合规性的更多信息，请参阅美国联邦信息处理标准（FIPS）140-2。

AWS Fargate FIPS-140 注意事项

在 Fargate 上使用 FIPS-140 合规性时，考虑以下各项：

- FIPS-140 合规性仅在 AWS GovCloud (US) 区域可用。
- FIPS-140 合规性默认关闭。您必须将其开启。
- 您的任务必须使用 FIPS-140 合规性的以下配置：
  - `operatingSystemFamily` 必须是 `LINUX`。
  - `cpuArchitecture` 必须是 `X86_64`。
  - Fargate 平台版本必须为 `1.4.0` 或更高版本。

在 Fargate 上使用 FIPS

请按照以下程序在 Fargate 上使用 FIPS-140 合规性。

1. 开启 FIPS-140 合规性。有关更多信息，请参阅 the section called “AWS Fargate 美国联邦信息处理标准（FIPS-140）合规性”。
2. 您可以选择使用 ECS Exec 运行以下命令来验证集群的 FIPS-140 合规性状态。

   将 *my-cluster* 替换为您集群的名称。

   返回值“1”表示您正在使用 FIPS。

   ```
   aws ecs execute-command --cluster cluster-name \
       --interactive \
       --command "cat /proc/sys/crypto/fips_enabled"
   ```

将 CloudTrail 用于 Fargate FIPS-140 审核

在您创建账户时，您的 AWS 账户中已启用 CloudTrail。当 Amazon ECS 中发生 API 和控制台活动时，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。

如要持续记录 AWS 账户中的事件（包括 Amazon ECS 事件），请创建跟踪记录，CloudTrail 将使用跟踪记录向 Amazon S3 存储桶传送日志文件。默认情况下，在控制台中创建跟踪记录时，此跟踪记录

应用于所有区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，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行动。有关更多信息，请参阅 the section called “使用 AWS CloudTrail 记录 Amazon ECS API 调用”。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `PutAccountSettingDefault` API 操作：


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIV5AJI5LXF5EXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/jdoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIPWIOFC3EXAMPLE",
    },
    "eventTime": "2023-03-01T21:45:18Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "PutAccountSettingDefault",
    "awsRegion": "us-gov-east-1",
    "sourceIPAddress": "52.94.133.131",
    "userAgent": "aws-cli/2.9.8 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
ecs.put-account-setting",
    "requestParameters": {
        "name": "fargateFIPSMode",
        "value": "enabled"
    },
    "responseElements": {
        "setting": {
            "name": "fargateFIPSMode",
            "value": "enabled",
            "principalArn": "arn:aws:iam::123456789012:user/jdoe"
        }
    },
    "requestID": "acdc731e-e506-447c-965d-f5f75EXAMPLE",
    "eventID": "6afced68-75cd-4d44-8076-0beEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
```

```
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "ecs-fips.us-gov-east-1.amazonaws.com"
    }
}
```

Amazon Elastic Container Service 的基础设施安全

作为一项托管式服务，Amazon Elastic Container Service 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 Amazon ECS。客户端必须支持以下内容：

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

您可以从任何网络位置调用这些 API 操作。Amazon ECS 支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制，因此请确保策略考虑网络位置的 IP 地址。您还可以使用 Amazon ECS 策略来控制来自特定 Amazon Virtual Private Cloud 端点或特定 VPC 的访问。事实上，这隔离了在 AWS 网络中仅从特定 VPC 到给定 Amazon ECS 资源的网络访问。有关更多信息，请参阅 Amazon ECS 接口 VPC 端点 (AWS PrivateLink)。

Amazon ECS 接口 VPC 端点 (AWS PrivateLink)

您可以将 Amazon ECS 配置为使用接口 VPC 端点以改善 VPC 的安全状况。接口端点由 AWS PrivateLink 提供支持，通过该技术，您可以使用私有 IP 地址私密访问 Amazon ECS API。AWS PrivateLink 限制您的 VPC 和 Amazon ECS 之间的所有网络流量到达 Amazon 网络。您无需互联网网关、NAT 设备或虚拟私有网关。

有关 AWS PrivateLink 和 VPC 端点的更多信息，请参阅 Amazon VPC 用户指南中的 VPC 端点。

注意事项

从 2023 年 12 月 23 日起推出的区域中的端点注意事项

在为 Amazon ECS 设置接口 VPC 端点之前，请注意以下事项：

- 您必须拥有以下区域特定的 VPC 端点：
  - `com.amazonaws.region.ecs-agent`
  - `com.amazonaws.region.ecs-telemetry`
  - `com.amazonaws.region.ecs`

  例如，加拿大西部（卡尔加里）（ca-west-1）区域需要以下 VPC 端点：
  - `com.amazonaws.ca-west-1.ecs-agent`
  - `com.amazonaws.ca-west-1.ecs-telemetry`
  - `com.amazonaws.ca-west-1.ecs`
- 使用模板在新区域中创建 AWS 资源并且该模板是从 2023 年 12 月 23 日之前推出的区域复制而来时，则根据具体的复制来源区域，请执行以下操作之一。

  例如，复制来源区域为美国东部（弗吉尼亚州北部）（us-east-1），复制目的地区域为加拿大西部（卡尔加里）（ca-west-1）。

配置	操作	
复制来源区域没有任何 VPC 端点。	为新区域创建所有三个 VPC 端点（例如 `com.amazonaws.ca-west-1.ecs-agent`）。	
复制来源区域包含区域特定的 VPC 端点。	a. 为新区域创建所有三个 VPC 端点（例如 `com.amazonaws.ca-west-1.ecs-agent`）。 b. 删除复制来源区域的所有三个 VPC 端点（例如 `com.amazo`	

配置	操作	
	naws.us-east-1.ecs-agent ）。	

Fargate 启动类型的 Amazon ECS VPC 端点的考虑因素

当 ecr.dkr 和 ecr.api 的 VPC 端点所在 VPC 与部署 Fargate 任务的 VPC 相同时，则将使用该 VPC 端点。如果没有 VPC 端点，则其将使用 Fargate 接口。

在为 Amazon ECS 设置接口 VPC 端点之前，请注意以下事项：

- 使用 Fargate 启动类型的任务不需要 Amazon ECS 的接口 VPC 端点，但您可能需要 Amazon ECR、Secrets Manager 或 Amazon CloudWatch Logs 的接口 VPC 端点，如以下几点所述。
  - 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 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的接口 VPC 端点 (AWS PrivateLink)。

    如果您的 VPC 没有互联网网关，您必须为 Amazon S3 创建网关端点。有关更多信息，请参阅《Amazon Elastic Container Registry 用户指南》中的创建 Amazon S3 网关端点。Amazon S3 的接口端点不能与 Amazon ECR 一起使用。

    > **Important**
    >
    > 如果您将 Amazon ECR 配置为使用接口 VPC 端点，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 端点的访问。有关更多信息，请参阅 Fargate 任务通过接口端点拉取 Amazon ECR 映像的权限。

  - 要允许您的任务从 Secrets Manager 拉取敏感数据，您必须为 Secrets Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的将 Secrets Manager 与 VPC 端点结合使用。
  - 如果您的 VPC 没有互联网网关，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 创建一个接口 VPC 端点。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的将 CloudWatch Logs 与接口 VPC 端点结合使用。
- VPC 端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端点。例如，假设您要在美国东部（弗吉尼亚州北部）运行任务。然后，您必须在美国东部（弗吉尼亚州北部）创建 Amazon ECS VPC 端点。在任何其他区域创建的 Amazon ECS VPC 端点都无法在美国东部（弗吉尼亚州北部）运行任务。

- VPC 端点仅通过 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 端点的安全组必须允许 TCP 端口 443 上来自 VPC 私有子网的传入连接。
- Envoy 代理的 Service Connect 管理使用 `com.amazonaws.region.ecs-agent` VPC 端点。当您不使用 VPC 端点时，Envoy 代理的 Service Connect 管理将使用该区域中的 `ecs-sc` 端点。有关每个区域的 Amazon ECS 端点的列表，请参阅 Amazon ECS 端点和配额。

EC2 启动类型的 Amazon ECS VPC 端点的考虑因素

在为 Amazon ECS 设置接口 VPC 端点之前，请注意以下事项：

- 使用 EC2 启动类型的任务要求启动它们的容器实例运行 `1.25.1` 或更高版本的 Amazon ECS 容器代理。有关更多信息，请参阅 Amazon ECS Linux 容器实例管理。
- 要允许您的任务从 Secrets Manager 拉取敏感数据，您必须为 Secrets Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的将 Secrets Manager 与 VPC 端点结合使用。
- 如果您的 VPC 没有互联网网关，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 创建一个接口 VPC 端点。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南中的将 CloudWatch Logs 与接口 VPC 端点结合使用。
- VPC 端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端点。例如，假设您要在美国东部（弗吉尼亚州北部）运行任务。然后，您必须在美国东部（弗吉尼亚州北部）创建 Amazon ECS VPC 端点。在任何其他区域创建的 Amazon ECS VPC 端点都无法在美国东部（弗吉尼亚州北部）运行任务。
- VPC 端点仅通过 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 端点的安全组必须允许 TCP 端口 443 上来自 VPC 私有子网的传入连接。

为 Amazon ECS 创建 VPC 端点

要为 Amazon ECS 服务创建VPC端点，请使用 Amazon VPC 用户指南中的创建接口端点过程创建以下端点。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，例如美国东部（俄亥俄）区域的 us-east-2。

ecs-agent 端点使用 ecs:poll API，而 ecs-telemetry 端点使用 ecs:poll 和 ecs:StartTelemetrySession API。

如果您当前有使用 EC2 启动类型的任务，则在创建 VPC 端点后，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。

重启 Amazon ECS 容器代理

1. 通过 SSH 登录到容器实例。
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 端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。

为 Amazon ECS 创建 VPC 端点策略

您可以为 VPC 端点附加控制对 Amazon ECS 的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

示例：Amazon ECS 操作的 VPC 端点策略

下面是用于 Amazon ECS 的端点策略示例。当附加到终端节点时，此策略会向您授予创建和列出集群的访问权限。`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 *。

```
{
   "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Amazon ECS 任务和容器安全性最佳实践

您应该将容器映像视为抵御“攻击”的第一道防线。不安全、构造不佳的映像可能允许攻击者逃离容器的边界并获得对主机的访问权限。您应该采取以下措施来缓解发生这种情况的风险。

在设置任务任务和容器时，建议执行以下操作。

创建最小的映像或使用 distroless 映像

首先从容器映像中移除所有多余的二进制文件。如果您使用的是来自 Amazon ECR Public Gallery 的不熟悉映像，请检查该映像以参考每个容器层的内容。您可以使用诸如 Dive 之类的应用程序来执行此操作。

或者，您可以使用仅包含您的应用程序及其运行时系统依赖项的 distroless 映像。它们不包含包管理器或 shell。Distroless 映像改善了“扫描仪的信噪比，减轻了根据需要确定来源的负担”。有关更多信息，请参阅有关 distroless 的 GitHub 文档。

Docker 有一种机制，可以从预留的、最小的镜像（称为 scratch）创建映像。有关更多信息，请参阅 Docker 文档中的使用 scratch 创建简单的父映像。使用像 Go 这样的语言，您可以创建一个静态链接的二进制文件，然后在 Dockerfile 中引用它。以下示例说明了如何完成此操作。

```
############################
# STEP 1 build executable binary
############################
FROM golang:alpine AS builder
# Install git.
# Git is required for fetching the dependencies.
RUN apk update && apk add --no-cache git
WORKDIR $GOPATH/src/mypackage/myapp/
COPY . .
# Fetch dependencies.
# Using go get.
RUN go get -d -v
# Build the binary.
RUN go build -o /go/bin/hello
############################
# STEP 2 build a small image
############################
FROM scratch
# Copy our static executable.
COPY --from=builder /go/bin/hello /go/bin/hello
# Run the hello binary.
ENTRYPOINT ["/go/bin/hello"]
This creates a container image that consists of your application and nothing else,
 making it extremely secure.
```

上一个示例也是多阶段构建示例。从安全角度来看，这些类型的构建很有吸引力，因为您可以使用它们来最大限度地减少推送到容器注册表的最终映像的大小。缺乏构建工具和其他无关二进制文件的容器映像通过减少映像的攻击面来改善您的安全状况。有关多阶段构建的更多信息，请参阅创建多阶段构建。

扫描您的映像中是否存在漏洞

与虚拟机映像类似，容器映像可能包含有漏洞的二进制文件和应用程序库，或者随着时间的推移而出现漏洞。防范漏洞的最佳方法是定期使用图像扫描仪扫描图像。

存储在 Amazon ECR 中的映像可以在推送时或需要时扫描（每 24 小时一次）。Amazon ECR 基本扫描使用一种开源图像扫描解决方案 Clair。Amazon ECR 增强扫描使用 Amazon Inspector。扫描图像后，结果将记录到 Amazon EventBridge 中的 Amazon ECR 事件流中。您还可以从 Amazon ECR

控制台中或通过调用 DescribeImageScanFindings API 来查看扫描结果。应删除或重建带有 HIGH 或 CRITICAL 漏洞的映像。如果已部署的映像出现漏洞，则应尽快将其更换。

Docker Desktop Edge 版本 2.3.6.0 或更高版本可以扫描本地映像。扫描由应用程序安全服务 Snyk 提供支持。当发现漏洞时，Snyk 会识别 Dockerfile 中存在漏洞的层次和依赖项。它还推荐了安全的替代方案，例如使用漏洞更少的更精简的基础映像，或者将特定的软件包升级到较新的版本。通过使用 Docker 扫描，开发人员可以在将映像推送到注册表之前解决潜在的安全问题。

- 使用 Amazon ECR 和 AWS Security Hub 自动实现映像合规性说明如何显示来自 AWS Security Hub 中的 Amazon ECR 的漏洞信息，以及如何通过阻止访问易受攻击的图像来自动修复。

移除映像的特殊权限

访问权限标志 setuid 和 setgid 允许以可执行文件所有者或组的权限运行可执行文件。从您的映像中移除具有这些访问权限的所有二进制文件，因为这些二进制文件可用于升级权限。考虑移除所有可能用于恶意目的 nc 和 curl 之类的 shell 和实用程序。您可以使用以下命令查找具有 setuid 和 setgid 访问权限的文件。

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

要从这些文件中移除这些特殊权限，请在您的容器映像中添加以下指令。

```
RUN find / -xdev -perm /6000 -type f -exec chmod a-s {} \; || true
```

创建一组精选映像

与其允许开发人员创建自己的映像，不如为组织中的不同应用程序堆栈创建一组经过审查的映像。通过这样做，开发人员可以放弃学习如何编写 Dockerfiles，而专注于编写代码。当更改合并到您的代码库中时，CI/CD 管道可以自动编译资产，然后将其存储在构件存储库中。最后，在将构件推送到 Docker 注册表（例如 Amazon ECR）之前，将其复制到相应的映像中。您至少应该创建一组基础映像，开发人员可以从中创建自己的 Dockerfiles。您应避免从 Docker Hub 中提取映像。您并不总是知道映像中有什么，在前 1000 个映像中，大约有五分之一存在漏洞。这些映像及其漏洞的列表可以在 https://vulnerablecontainers.org/ 中找到。

扫描应用程序包和库中是否存在漏洞

现在，开源库已普遍使用。与操作系统和操作系统包一样，这些库可能存在漏洞。作为开发生命周期的一部分，当发现严重漏洞时，应对这些库进行扫描和更新。

Docker Desktop 使用 Snyk 执行本地扫描。它还可用于查找开源库中的漏洞和潜在的许可问题。可以直接将其集成到开发人员工作流程中，从而降低开源库带来的风险。有关更多信息，请参阅以下主题：

- 开源应用程序安全工具包括了用于检测应用程序漏洞的工具列表。

执行静态代码分析

在构建容器映像之前，您应该执行静态代码分析。它是针对您的源代码执行的，用于识别代码错误和可能被恶意行为者利用的代码，例如错误注入。SonarQube 是静态应用程序安全测试（SAST）的常用选项，它支持多种不同的编程语言。

以非根用户身份运行容器

您应该以非根用户身份运行容器。默认情况下，除非 USER 指令包含在您的 Dockerfile 中，否则容器将以 `root` 用户身份运行。Docker 分配的默认 Linux 功能限制了可以作为 `root` 身份运行的操作，但只能稍作限制。例如，以 `root` 身份运行的容器仍不允许访问设备。

作为 CI/CD 管道的一部分，您应该使用 lint Dockerfiles 来查找 USER 指令，如果缺少该指令，则构建失败。有关更多信息，请参阅以下主题：

- Dockerfile-lint 是 RedHat 推出的开源工具，可用于检查文件是否符合最佳实践。
- Hadolint 是另一种用于构建符合最佳实践的 Docker 映像的工具。

使用只读的根文件系统

您应该使用只读的根文件系统。默认情况下，容器的根文件系统是可编写的。当您为容器配置一个 RO（只读）根文件系统时，它会强制您明确定义数据的保存位置。这可以减少攻击面，因为除非特别授予权限，否则无法写入容器的文件系统。

> **Note**
>
> 拥有只读的根文件系统可能会导致某些期望能够写入文件系统的操作系统包出现问题。如果您打算使用只读的根文件系统，请事先进行全面测试。

使用 CPU 和内存限制配置任务（Amazon EC2）

您应该为任务配置 CPU 和内存限制，以最大限度地降低以下风险。针对任务中所有容器可以预留的 CPU 和内存量，任务的资源限制设定了上限。如果未设置限制，则任务可以访问主机的 CPU 和内存。这可能会导致在共享主机上部署的任务使其他任务耗尽系统资源的问题。

> **Note**
>
> AWS Fargate 任务上的 Amazon ECS 要求您指定 CPU 和内存限制，因为它会将这些值用于计费。对于 Amazon ECS Fargate 来说，一项占用所有系统资源的任务不是问题，因为每个任务都是在自己的专用实例上运行的。如果您未指定内存限制，Amazon ECS 会为每个容器分配至少 4MB 的存储空间。同样，如果没有为任务设置 CPU 限制，Amazon ECS 容器代理会为其分配至少 2 个 CPU。

在 Amazon ECR 中使用不可变标签

借助 Amazon ECR，您可以而且应该使用具有不可变标签的配置映像。这样可以防止将修改或更新的映像版本推送到带有相同标签的映像存储库。这样可以防止攻击者在带有相同标签的映像上推送受损版本的映像。通过使用不可变标签，您可以有效地强迫自己为每次更改推送一个带有不同标签的新映像。

避免以特权身份运行容器（Amazon EC2）

您应该避免以特权身份运行容器。对于后台，以 `privileged` 身份运行的容器在主机上以扩展权限运行。这意味着容器会继承分配给主机上的 `root` 的所有 Linux 功能。应严格限制或禁止其使用。建议将 Amazon ECS 容器代理环境变量 `ECS_DISABLE_PRIVILEGED` 设置为 `true`，以防止容器在不需要 `privileged` 时在特定主机以 `privileged` 身份运行。或者，您可以使用 AWS Lambda 扫描任务定义以了解 `privileged` 参数的使用情况。

> **Note**
>
> AWS Fargate 上的 Amazon ECS 不支持容器以 `privileged` 身份运行。

从容器中移除不必要的 Linux 功能

以下是分配给 Docker 容器的默认 Linux 功能的列表。有关每项功能的更多信息，请参阅 Linux 功能概述。

```
CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FOWNER, CAP_FSETID, CAP_KILL,
CAP_SETGID, CAP_SETUID, CAP_SETPCAP, CAP_NET_BIND_SERVICE,
CAP_NET_RAW, CAP_SYS_CHROOT, CAP_MKNOD, CAP_AUDIT_WRITE,
CAP_SETFCAP
```

如果容器不需要上面列出的所有 Docker 内核功能，请考虑将其从容器中删除。有关 Docker 内核每项功能的更多信息，请参阅 KernelCapabilities。您可以通过执行以下操作查找哪些功能在使用中：

- 安装操作系统包 libcap-ng 并运行 `pscap` 实用程序以列出每个进程正在使用的功能。
- 您还可以使用 capsh 来解密进程正在使用哪些功能。

使用客户自主管理型密钥（CMK）加密推送到 Amazon ECR 的映像

您应该使用客户自主管理型密钥（CMK）加密推送到 Amazon ECR 的映像。推送到 Amazon ECR 的映像会使用 AWS Key Management Service（AWS KMS）托管密钥自动进行静态加密。如果您更愿意使用自己的密钥，Amazon ECR 现在支持使用客户自主管理型密钥（CMK）进行 AWS KMS 加密。在使用 CMK 启用服务器端加密之前，请查看静态加密文档中列出的注意事项。

Amazon ECS 教程

以下教程说明在使用 Amazon ECS 时如何执行常见任务。

您可以借助以下任一教程，使用 AWS CLI 在 Amazon ECS 上部署任务

教程概述	了解更多信息
创建 Fargate 启动类型的 Linux 任务。	使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Linux 任务
创建 Fargate 启动类型的 Windows 任务。	使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Windows 任务
创建 EC2 启动类型的 Linux 任务。	使用 AWS CLI 创建 EC2 启动类型的 Amazon ECS 任务

您可以借助以下任一教程来详细了解监控和日志记录。

教程概述	了解更多信息
设置一个简单的 Lambda 函数，用于侦听任务事件并将其写出到 CloudWatch Logs 日志流。	配置 Amazon ECS 以侦听 CloudWatch Events 事件
配置一个 Amazon EventBridge 事件规则，用于仅捕获任务因某个主要容器终止而停止的任务事件。	针对 Amazon ECS 任务停止事件发送 Amazon Simple Notification Service 警报
连接原属于一个上下文但被分割为多个记录或日志行的日志消息。	连接多行或堆栈跟踪 Amazon ECS 日志消息

教程概述	了解更多信息	
在 Amazon ECS 中运行的 Windows 实例上部署 Fluent Bit 容器，以便将 Windows 任务生成的日志流式传输到 Amazon CloudWatch 进行集中日志记录。	在 Amazon ECS Windows 容器上部署 Fluent Bit	

您可以借助以下任一教程来详细了解如何在 Amazon ECS 上将 Active Directory 身份验证与组托管服务账户结合使用。

教程概述	了解更多信息	
在 EC2 上将组托管服务账户与 Linux 容器结合使用。	将 gMSA 用于 Amazon ECS 上的 EC2 Linux 容器	
在 EC2 上将组托管服务账户与 Windows 容器结合使用。	了解了解如何将 gMSA 用于适用于 Amazon ECS 的 EC2 Windows 容器	
在 Fargate 上将组托管服务账户与 Linux 容器结合使用。	将 gMSA 用于 Fargate 上的 Linux 容器	
创建一个运行 Windows 容器的任务，该容器含有使用无域组托管服务账户访问 Active Directory 所需的凭证。	通过 AWS CLI 将 Amazon ECS Windows 容器与无域 gMSA 结合使用	

使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Linux 任务

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 Linux 任务和执行其他常见方案。使用最新版本的 AWS CLI。有关如何升级到最新版本的更多信息，请参阅安装 AWS Command Line Interface。

主题

- 先决条件
- 步骤 1：创建集群
- 第 2 步：注册 Linux 任务定义
- 第 3 部：列出任务定义
- 步骤 4：创建服务
- 步骤 5：列出服务
- 步骤 6：描述正在运行的服务
- 步骤 7：测试
- 步骤 8：清除

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 Command Line Interface。
- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。本教程使用的是托管在 Amazon ECR Public 上的容器映像，因此，您的任务必须具有互联网访问权限。要让您的任务连接到互联网，请使用下列选项之一。
  - 将私有子网与具有弹性 IP 地址的 NAT 网关结合使用。
  - 使用公有子网并向任务分配公有 IP 地址。

  有关更多信息，请参阅 the section called “创建 Virtual Private Cloud”。

  有关安全组的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的 VPC 的默认安全组和示例规则。
- 如果您使用私有子网学习本教程，则可以使用 Amazon ECS Exec 直接与您的容器交互并测试部署。您需要创建一个任务 IAM 角色才能使用 ECS Exec。有关任务 IAM 角色和其他先决条件的更多信息，请参阅使用 Amazon ECS Exec 进行调试。
- （可选）AWS CloudShell 是一种为客户提供命令行的工具，而无需创建自己的 EC2 实例。有关更多信息，请参阅AWS CloudShell《用户指南》中的什么是AWS CloudShell？。

步骤 1：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` 群集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",
        "defaultCapacityProviderStrategy": [],
        "statistics": [],
        "capacityProviders": [],
        "tags": [],
        "clusterName": "fargate-cluster",
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

第 2 步：注册 Linux 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，它使用托管在 Docker Hub 上的 httpd 容器映像创建 PHP Web 应用程序。有关可用任务定义参数的更多信息，请参阅 Amazon ECS 任务定义。在本教程中，taskRoleArn 只有当您再私有子网中部署任务并希望测试部署时才需要。将 taskRoleArn 替换为您创建的 IAM 任务角色，以便使用 ECS Exec，如 先决条件 中所述。

```
{
       "family": "sample-fargate",
       "networkMode": "awsvpc",
       "taskRoleArn": "arn:aws:iam::aws_account_id:role/execCommandRole",
       "containerDefinitions": [
           {
               "name": "fargate-app",
               "image": "public.ecr.aws/docker/library/httpd:latest",
               "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"
```

```
}
```

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。

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 命令在完成其注册后会返回任务定义的描述。

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"
    ]
}
```

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，您将使用集群中运行的一个 `sample-fargate:1` 任务定义实例来创建服务。此任务需要连接到互联网，您可以通过两种方法实现这一点。一种方法是将使用 NAT 网关配置的私有子网与公有子网中的弹性 IP 地址结合使用。另一种方法是使用公有子网并向任务分配公有 IP 地址。下面提供了这两种方法的示例。

使用私有子网的示例。需要 `enable-execute-command` 选项才能使用 Amazon ECS Exec。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}" --enable-execute-command
```

使用公有子网的示例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令在完成其注册后会返回任务定义的描述。

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-cluster/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
```

如果成功，这将返回服务失败和服务的描述。例如，在 `services` 部分中，您将找到有关部署的信息，例如任务的正在运行或待处理状态。也可以找到有关任务定义、网络配置和带时间戳的事件的信息。在失败部分中，您将找到与调用关联的失败的信息（如果有）。对于问题排查，请参阅服务事件消息。有关服务描述的更多信息，请参阅描述服务。

```
{
    "services": [
        {
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
```

```
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "ENABLED"
                }
            },
            "launchType": "FARGATE",
            "enableECSManagedTags": false,
            "loadBalancers": [],
            "deploymentController": {
                "type": "ECS"
            },
            "desiredCount": 1,
            "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
            },
            "createdAt": 1692283199.771,
            "schedulingStrategy": "REPLICA",
            "placementConstraints": [],
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "ENABLED"
                        }
                    },
                    "pendingCount": 0,
                    "launchType": "FARGATE",
                    "createdAt": 1692283199.771,
                    "desiredCount": 1,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate:1",
                    "updatedAt": 1692283199.771,
```

```
                    "platformVersion": "1.4.0",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "serviceName": "fargate-service",
            "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,
            "status": "ACTIVE",
            "serviceRegistries": [],
            "pendingCount": 0,
            "createdBy": "arn:aws:iam::aws_account_id:user/user_name",
            "platformVersion": "LATEST",
            "placementStrategy": [],
            "propagateTags": "NONE",
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1"
        }
    ],
    "failures": []
}
```

步骤 7：测试

使用公有子网部署的测试任务

描述服务中的任务，以便您可以获得该任务的弹性网络接口（ENI）。

首先，获取任务 ARN。

```
aws ecs list-tasks --cluster fargate-cluster --service fargate-service
```

输出包含任务 ARN。

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE
    ]
}
```

描述任务并找到 ENI ID。将任务 ARN 用于 tasks 参数。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/service/EXAMPLE
```

输出中列出了附件信息。

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
```

```
                    ]
                }
…
}
```

请描述 ENI 以获取公有 IP 地址。

```
aws ec2 describe-network-interfaces --network-interface-id  eni-0fa40520aeEXAMPLE
```

公有 IP 地址在输出中。

```
{
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-229-42-222.compute-1.amazonaws.com",
                "PublicIp": "198.51.100.2"
            },
…
}
```

在 Web 浏览器中，输入公有 IP 地址，您应该可以看到显示 Amazon ECS 样本应用程序的网页。

使用私有子网部署的测试任务

描述任务并找到 managedAgents 以验证 ExecuteCommandAgent 是否正在运行。记下 privateIPv4Address 以供将来使用。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/fargate-service/EXAMPLE
```

输出中列出了托管代理信息。

```
{
     "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
```

```
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.143.156"
                        }
                    ]
                }
            ],
    ...
    "containers": [
        {
        ...
       "managedAgents": [
                        {
                            "lastStartedAt": "2023-08-01T16:10:13.002000+00:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                ],
        ...
    }
```

验证 `ExecuteCommandAgent` 是否正在运行后，可以运行以下命令，以在任务中的容器上运行交互式 shell。

```
aws ecs execute-command --cluster fargate-cluster \
    --task  arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE  \
    --container  fargate-app \
    --interactive \
    --command "/bin/sh"
```

交互式 shell 运行后，运行以下命令来安装 cURL。

```
apt update
```

```
apt install curl
```

安装 cURL 后，使用之前获得的私有 IP 地址运行以下命令。

```
 curl 10.0.143.156
```

您应该会看到与 Amazon ECS 示例应用程序网页等效的 HTML。

```
<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>
```

步骤 8：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。

删除服务。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

请删除集群。

```
aws ecs delete-cluster --cluster fargate-cluster
```

使用 AWS CLI 创建 Fargate 启动类型的 Amazon ECS Windows 任务

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 Windows 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 Command Line Interface。

主题

- 先决条件
- 步骤 1：创建集群
- 步骤 2：注册 Windows 任务定义
- 步骤 3：列出任务定义
- 步骤 4：创建服务
- 步骤 5：列出服务
- 步骤 6：描述正在运行的服务
- 步骤 7：清理

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 Command Line Interface。
- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 您已创建要使用的 VPC 和安全组。本教程使用的是托管在 Docker Hub 上的容器映像，因此，您的任务必须具有互联网访问权限。要让您的任务连接到互联网，请使用下列选项之一。
  - 将私有子网与具有弹性 IP 地址的 NAT 网关结合使用。
  - 使用公有子网并向任务分配公有 IP 地址。

  有关更多信息，请参阅 the section called “创建 Virtual Private Cloud”。

  有关安全组的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的 VPC 的默认安全组和示例规则。

- （可选）AWS CloudShell 是一种为客户提供命令行的工具，而无需创建自己的 EC2 实例。有关更多信息，请参阅AWS CloudShell《用户指南》中的什么是AWS CloudShell？。

步骤 1：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` 群集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"
    }
}
```

步骤 2：注册 Windows 任务定义

您必须先注册任务定义才能在 Amazon ECS 集群上运行 Windows 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 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。

```
{
```

```
    "containerDefinitions": [
        {
            "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"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

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。

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 命令在完成其注册后会返回任务定义的描述。

步骤 3：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-windows:1"
    ]
}
```

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，您将使用集群中运行的一个 `sample-fargate:1` 任务定义实例来创建服务。此任务需要连接到互联网，您可以通过两种方法实现这一点。一种方法是将使用 NAT 网关配置的私有子网与公有子网中的弹性 IP 地址结合使用。另一种方法是使用公有子网并向任务分配公有 IP 地址。下面提供了这两种方法的示例。

使用私有子网的示例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234]}"
```

使用公有子网的示例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令在完成其注册后会返回任务定义的描述。

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
```

如果成功，这将返回服务失败和服务的描述。例如，在服务部分中，您将找到有关部署的信息，例如任务的正在运行或待处理状态。也可以找到有关任务定义、网络配置和带时间戳的事件的信息。在失败部分中，您将找到与调用关联的失败的信息（如果有）。对于问题排查，请参阅服务事件消息。有关服务描述的更多信息，请参阅描述服务。

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate-windows: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-windows: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": []
}
```

步骤 7：清理

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。

删除服务。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

请删除集群。

```
aws ecs delete-cluster --cluster fargate-cluster
```

使用 AWS CLI 创建 EC2 启动类型的 Amazon ECS 任务

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。使用最新版本的 AWS CLI。有关如何升级到最新版本的更多信息，请参阅安装 AWS Command Line Interface。

主题

- 先决条件
- 步骤 1：创建集群
- 步骤 2：使用 Amazon ECS AMI 启动实例
- 第 3 步：列出容器实例
- 第 4 步：描述容器实例
- 第 5 步：注册任务定义
- 第 6 部：列出任务定义
- 第 7 步：运行任务
- 第 8 步：列出任务
- 第 9 步：描述正在运行的任务

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 Command Line Interface。
- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 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 the section called “创建 Virtual Private Cloud”。
- （可选）AWS CloudShell 是一种为客户提供命令行的工具，而无需创建自己的 EC2 实例。有关更多信息，请参阅AWS CloudShell《用户指南》中的什么是AWS CloudShell？。

步骤 1：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` 群集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"
    }
}
```

步骤 2：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 Linux 容器实例以了解更多信息。

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。

```
{
    "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": []
                }
            ]
        }
    ]
}
```

配置 Amazon ECS 以侦听 CloudWatch Events 事件

了解如何设置一个简单的 Lambda 函数，用于侦听任务事件并将其写出到 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 the section called “为 Fargate 启动类型创建集群” 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 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
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 的版本，例如 Python 3.9。
    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 3.9 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. 选择保存。

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 Management Console 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 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
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。预设情况下，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（配置详细信息）。
6. 对于 Rule definition，键入规则的名称和说明，然后选择 Create rule。

步骤 3：创建任务定义

创建任务定义。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task Definitions。
3. 选择 Create new Task Definition（创建新的任务定义）、Create new revision with JSON（使用 JSON 创建新的修订）。
4. 将以下示例任务定义复制并粘贴到框中，然后选择 Save (保存)。

```
{
    "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,
            "memory": 300,
            "image": "httpd:2.4",
```

```
            "name": "simple-app"
        }
    ],
    "family": "console-sample-app-static"
}
```

5. 选择创建。

步骤 4：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/v2 打开控制台。
2. 选择 Task definitions（任务定义）。
3. 选择 console-sample-app-static，然后选择 Deploy（部署），Run new task（运行新任务）。
4. 对于 Cluster（集群），选择默认值，然后选择 Deploy（部署）。
5. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
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。

针对 Amazon ECS 任务停止事件发送 Amazon Simple Notification Service 警报

配置一个 Amazon EventBridge 事件规则，用于仅捕获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，请执行使用 AWS Fargate 上的 Linux 容器开始使用控制台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 EventBridge 规则。

先决条件：为 Amazon SNS 配置权限

如要允许 EventBridge 发布到 Amazon SNS 主题，请使用 aws sns get-topic-attributes 和 aws sns set-topic-attributes 命令。

有关如何添加权限的信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 权限。

添加以下权限：

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sns: Publish",
  "Resource": "arn:aws:sns:region:account-id:TaskStoppedAlert",
}
```

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。

有关如何创建和订阅 Amazon SNS 主题的信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 入门，并使用下表确定选择哪些选项。

选项	值
类型	Standard
名称	TaskStoppedAlert
协议	Email
终端节点	您当前有权访问的电子邮件地址

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。

有关如何创建和订阅 Amazon SNS 主题的信息，请参阅《Amazon EventBridge 用户指南》中的在 Amazon EventBridge 中创建规则，并使用以下表格确定要选择哪些选项。

选项	值
Rule type	具有事件模式的规则
事件源	AWS 事件或 EventBridge 合作伙伴事件
事件模式	自定义模式（JSON 编辑器）

事件模式	<pre>{
	  "source":[
	  "aws.ecs"
	  ],
	  "detail-type":[
	  "ECS Task State Change"
	  ],
	  "detail":{
	  "lastStatus":[
	  "STOPPED"
	  ],
	  "stoppedReason":[
	  "Essential container in task exited"
	  ]
	  }
	}</pre>

Target type	AWS 服务
目标	SNS 主题

选项	值	
主题	TaskStoppedAlert（您在步骤 1 中创建的主题）	

步骤 3：测试您的规则

通过运行在启动后不久退出的任务来验证规则是否有效。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。如果您具有可满足规则要求的现有任务定义，请使用该定义运行任务。如果您不具有该定义，以下步骤将引导您注册 Fargate 任务定义并运行它。

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在导航窗格中，选择 Task definitions（任务定义）。
3. 选择 Create new task definition（创建新的任务定义）、Create new task definition with JSON（使用 JSON 创建新的任务定义）。
4. 在 JSON 编辑器框中，编辑您的 JSON 文件，将以下内容复制到编辑器中。

```
{
   "containerDefinitions":[
      {
         "command":[
            "sh",
            "-c",
            "sleep 5"
         ],
         "essential":true,
         "image":"amazonlinux:2",
         "name":"test-sleep"
      }
   ],
   "cpu":"256",
   "executionRoleArn":"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family":"fargate-task-definition",
   "memory":"512",
   "networkMode":"awsvpc",
   "requiresCompatibilities":[
      "FARGATE"
   ]
}
```

5. 选择创建。

从控制台运行任务

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 在集群页面上，选择您在先决条件中创建的集群。
3. 从任务选项卡上，选择运行新任务。
4. 对于应用程序类型，选择任务。
5. 对于任务定义，选择 fargate-task-definition。
6. 对于 Desired tasks（预期任务），请输入要启动的任务数量。
7. 选择创建。

连接多行或堆栈跟踪 Amazon ECS 日志消息

从 AWS for Fluent Bit 版本 2.22.0 开始，包含多行筛选条件。多行筛选条件有助于连接原属于一个上下文但被分割为多个记录或日志行的日志消息。有关多行筛选条件的更多信息，请参阅 Fluent Bit 文档。

拆分日志消息的常见示例有：

- 堆栈跟踪。
- 在多行上打印日志的应用程序。
- 因为比指定的运行时间最大缓冲区大小长而拆分的日志消息。您可以按照 GitHub 中的以下示例来连接被容器运行时拆分的日志消息：FireLens Example: Concatenate Partial/Split Container Logs（FireLens 示例：连接部分/拆分的容器日志）。

所需的 IAM 权限

您拥有使用容器代理从 Amazon ECR 中提取容器镜像以及使用容器将日志路由到 CloudWatch Logs 的所需 IAM 权限。

对于这些权限，您还必须具有以下角色：

- 任务 IAM 角色。
- 任务执行 IAM 角色。

使用 JSON 策略编辑器创建策略

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 在页面的顶部，选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 输入以下 JSON 策略文档：

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

6. 选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅《IAM 用户指南》中的调整策略结构。

7. 在查看并创建页面上，为您要创建的策略输入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

确定何时使用多行日志设置

以下是您将在使用默认日志设置的 CloudWatch Logs 控制台中看到的示例日志片段。您可以查看以 log 开头的行，以确定是否需要多行筛选条件。当上下文相同时，您可以使用多行日志设置，在本例中，上下文为“com.myproject.model.MyProject”。

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-
app", "source=": "stdout", "log": ": "     at com.myproject.modele.
(MyProject.badMethod.java:22)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at com.myproject.model.MyProject.badMethod(MyProject.java:22)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app", "stdout",
 "log": ": "     at com.myproject.modele.(MyProject.oneMoreMethod.java:18)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at
 com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE,
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

使用多行日志设置后，输出将与以下示例类似。

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app",
 "stdout",...
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log:    "September 20, 2022 06:41:48 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
    at com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)
 com.myproject.module.MyProject.main(MyProject.java:6)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:2"
     }
```

解析和连接选项

要解析日志并连接因换行符而被拆分的行，您可以使用这两个选项之一。

- 使用自己的解析器文件，该文件中包含用于解析和连接属于同一消息的行的规则。
- 使用 Fluent Bit 内置解析器。有关 Fluent Bit 内置解析器支持的语言的列表，请参阅 Fluent Bit 文档。

以下教程将引导您完成每个使用案例的步骤。这些步骤向您展示了如何连接多行并将日志发送到 Amazon CloudWatch。您可以为日志指定其他目标。

示例：使用您创建的解析器

在本示例中，您将完成以下步骤：

1. 为 Fluent Bit 容器生成并上传镜像。
2. 为运行、失败和生成多行堆栈跟踪的演示多行应用程序生成并上传镜像。
3. 创建任务定义并运行任务。
4. 查看日志以验证跨越多行的消息是否连接起来。

为 Fluent Bit 容器生成并上传镜像

此镜像将包括您在其中指定正则表达式的解析器文件和引用解析器文件的配置文件。

1. 使用名称 FluentBitDockerImage 创建文件夹。
2. 在文件夹内，创建解析器文件，其中包含用于解析日志和连接属于同一消息的行的规则。

   a. 将以下内容粘贴到解析器文件中：

```
[MULTILINE_PARSER]
    name          multiline-regex-test
    type          regex
    flush_timeout 1000
    #
    # Regex rules for multiline parsing
    # ---------------------------------
    #
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules |   state name  | regex pattern                  | next state
    # ------|---------------|--------------------------------------------
    rule      "start_state"   "/(Dec \d+ \d+\:\d+\:\d+)(.*)/"  "cont"
    rule      "cont"          "/^\s+at.*/"                     "cont"
```

   自定义正则表达式模式时，我们建议您使用正则表达式编辑器来测试表达式。

   b. 将该文件保存为 parsers_multiline.conf。

3. 在 FluentBitDockerImage 文件夹中，创建引用您在上一步中创建的解析器文件的自定义配置文件。

   有关自定义配置文件的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的指定自定义配置文件

   a. 将以下内容粘贴到该文件中：

```
[SERVICE]
    flush                1
    log_level            info
    parsers_file         /parsers_multiline.conf
```

```
[FILTER]
    name                  multiline
    match                 *
    multiline.key_content log
    multiline.parser      multiline-regex-test
```

> **Note**
>
> 您必须使用解析器的绝对路径。

b. 将该文件保存为 `extra.conf`。

4. 在 `FluentBitDockerImage` 文件夹中，使用 Fluent Bit 镜像以及您创建的解析器和配置文件创建 Dockerfile。

a. 将以下内容粘贴到该文件中：

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest

ADD parsers_multiline.conf /parsers_multiline.conf
ADD extra.conf /extra.conf
```

b. 将该文件保存为 `Dockerfile`。

5. 使用 Dockerfile，构建一个包含解析器和自定义配置文件的自定义 Fluent Bit 镜像。

> **Note**
>
> 您可以将解析器文件和配置文件放置在 Docker 镜像中的任何位置，但 `/fluent-bit/etc/fluent-bit.conf` 除外，因为 FireLens 使用这个文件路径。

a. 生成镜像：`docker build -t fluent-bit-multiline-image .`

其中：`fluent-bit-multiline-image` 是此示例中镜像的名称。

b. 验证是否已正确创建镜像：`docker images –filter reference=fluent-bit-multiline-image`

如果成功，输出将显示该镜像和 `latest` 标签。

6. 将自定义 Fluent Bit 镜像上传到 Amazon Elastic Container Registry。

a. 创建用于存储镜像的 Amazon ECR 存储库：`aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

   其中：`fluent-bit-multiline-repo` 是存储库的名称，`us-east-1` 是此示例中的区域。

   输出为您提供了新存储库的详细信息。

b. 使用上一个输出中的 `repositoryUri` 值标记您的镜像：`docker tag fluent-bit-multiline-image repositoryUri`

   例如：`docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

c. 运行 Docker 镜像以验证它是否正确运行：`docker images –filter reference=repositoryUri`

   在输出中，存储库名称从 fluent-bit-multiline-repo 更改为 `repositoryUri`。

d. 通过运行 `aws ecr get-login-password` 命令并指定您要对其进行身份验证的注册表 ID 对 Amazon ECR 进行身份验证：`aws ecr get-login-password | docker login --username AWS --password-stdin registry ID.dkr.ecr.region.amazonaws.com`

   例如：`ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

   此时会显示成功登录消息。

e. 将镜像推送到 Amazon ECR：`docker push registry ID.dkr.ecr.region.amazonaws.com/repository name`

   例如：`docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

为演示多行应用程序生成并上传镜像

此镜像将包括运行应用程序的 Python 脚本文件和示例日志文件。

运行任务时，应用程序会模拟运行，然后失败并创建堆栈跟踪。

1. 创建名为 `multiline-app` 的文件夹：`mkdir multiline-app`
2. 创建 Python 脚本文件。

a. 在 multiline-app 文件夹中，创建一个文件并将其命名为 main.py。

b. 将以下内容粘贴到该文件中：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

c. 保存 main.py 文件。

3. 创建示例日志文件。

a. 在 multiline-app 文件夹中，创建一个文件并将其命名为 test.log。

b. 将以下内容粘贴到该文件中：

```
single line...
Dec 14 06:41:08 Exception in thread "main" java.lang.RuntimeException:
 Something has gone wrong, aborting!
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)
    at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
    at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)
    at com.myproject.module.MyProject.someMethod(MyProject.java:10)
    at com.myproject.module.MyProject.main(MyProject.java:6)
another line...
```

c. 保存 test.log 文件。

4. 在 multiline-app 文件夹中，创建 Dockerfile。

a. 将以下内容粘贴到该文件中：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 保存 Dockerfile 文件。

5. 使用 Dockerfile 生成镜像。

a. 生成镜像：`docker build -t multiline-app-image .`

其中：`multiline-app-image` 是此示例中镜像的名称。

b. 验证是否已正确创建镜像：`docker images –filter reference=multiline-app-image`

如果成功，输出将显示该镜像和 `latest` 标签。

6. 将镜像上载到 Amazon Elastic Container 注册表。

a. 创建用于存储镜像的 Amazon ECR 存储库：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

其中：`multiline-app-repo` 是存储库的名称，`us-east-1` 是此示例中的区域。

输出为您提供了新存储库的详细信息。记下 `repositoryUri` 值，您将需要在后续步骤中使用该值。

b. 使用上一个输出中的 `repositoryUri` 值标记您的镜像：`docker tag *multiline-app-image repositoryUri*`

例如：`docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. 运行 Docker 镜像以验证它是否正确运行：`docker images –filter reference=*repositoryUri*`

在输出中，存储库名称从 multiline-app-repo 更改为 repositoryUri 值。

d. 将镜像推送到 Amazon ECR：docker push *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*repository name*

例如：docker push *xxxxxxxxxxxx*.dkr.ecr.*us-east-1*.amazonaws.com/multiline-app-repo

创建任务定义并运行任务

1. 使用文件名 multiline-task-definition.json 创建任务定义文件。
2. 将以下内容粘贴到 multiline-task-definition.json 文件中：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN,
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
```

```
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

替换 multiline-task-definition.json 任务定义中的以下内容：

a. *task role ARN*

要查找任务角色 ARN，请转到 IAM 控制台。选择 Roles（角色），然后查找您创建的 ecs-task-role-for-firelens 任务角色。选择角色，然后复制 Summary（摘要）部分中显示的 ARN。

b. *execution role ARN*

要查找执行角色 ARN，请转到 IAM 控制台。选择 Roles（角色），然后查找 ecsTaskExecutionRole 角色。选择角色，然后复制 Summary（摘要）部分中显示的 ARN。

c. *aws_account_id*

要查找您的 aws_account_id，登录 AWS Management Console。在右上角选择您的用户名，然后复制您的账户 ID。

d. *us-east-1*

如有必要，请更换区域。

3. 注册任务定义文件：aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region *region*
4. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
5. 在导航窗格中，选择 Task Definitions（任务定义），然后选择 firelens-example-multiline 系列，因为我们已在上面任务定义的第一行中将任务定义注册到这个系列。

6. 选择最新版本。
7. 选择部署、运行任务。
8. 在运行任务页面上，对于集群，选择集群，然后在联网下的子网中，选择任务的可用子网。
9. 选择创建。

验证 Amazon CloudWatch 中的多行日志消息是否已连接

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 从导航窗格中，展开 Logs（日志）并选择 Log groups（日志组）。
3. 选择 `multiline-test/applicatio` 日志组。
4. 选择日志。查看消息。与解析器文件中的规则匹配的行将串联起来，并显示为单条消息。

   以下日志片段显示了在单个 Java 堆栈跟踪事件中连接的行：

```
{
    "container_id": "xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
 at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
 com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)\n
 at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)\n
 at com.myproject.module.MyProject.someMethod(MyProject.java:10)\n    at
 com.myproject.module.MyProject.main(MyProject.java:6)",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

   以下日志代码段显示了如果您运行的是未配置为连接多行日志消息的 Amazon ECS 容器，则如何只用一行显示相同的消息。

```
{
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
```

```
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
}
```

示例：使用 Fluent Bit 内置解析器

在本示例中，您将完成以下步骤：

1. 为 Fluent Bit 容器生成并上传镜像。
2. 为运行、失败和生成多行堆栈跟踪的演示多行应用程序生成并上传镜像。
3. 创建任务定义并运行任务。
4. 查看日志以验证跨越多行的消息是否连接起来。

为 Fluent Bit 容器生成并上传镜像

此镜像将包含一个引用 Fluent Bit 解析器的配置文件。

1. 使用名称 `FluentBitDockerImage` 创建文件夹。
2. 在 `FluentBitDockerImage` 文件夹中，创建引用 Fluent Bit 内置解析器文件的自定义配置文件。

   有关自定义配置文件的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的指定自定义配置文件

   a. 将以下内容粘贴到该文件中：

   ```
   [FILTER]
       name                  multiline
       match                 *
       multiline.key_content log
       multiline.parser      go
   ```

   b. 将该文件保存为 `extra.conf`。
3. 在 `FluentBitDockerImage` 文件夹中，使用 Fluent Bit 镜像以及您创建的解析器和配置文件创建 Dockerfile。

   a. 将以下内容粘贴到该文件中：

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
ADD extra.conf /extra.conf
```

b. 将该文件保存为 Dockerfile。

4. 使用 Dockerfile，构建一个包含自定义配置文件的自定义 Fluent Bit 镜像。

> **Note**
>
> 您可以将配置文件放置在 Docker 镜像中的任何位置，但 /fluent-bit/etc/fluent-bit.conf 除外，因为 FireLens 使用这个文件路径。

a. 生成镜像：`docker build -t fluent-bit-multiline-image .`

其中：`fluent-bit-multiline-image` 是此示例中镜像的名称。

b. 验证是否已正确创建镜像：`docker images –filter reference=fluent-bit-multiline-image`

如果成功，输出将显示该镜像和 `latest` 标签。

5. 将自定义 Fluent Bit 镜像上传到 Amazon Elastic Container Registry。

a. 创建用于存储镜像的 Amazon ECR 存储库：`aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

其中：`fluent-bit-multiline-repo` 是存储库的名称，`us-east-1` 是此示例中的区域。

输出为您提供了新存储库的详细信息。

b. 使用上一个输出中的 `repositoryUri` 值标记您的镜像：`docker tag` *`fluent-bit-multiline-image repositoryUri`*

例如：`docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

c. 运行 Docker 镜像以验证它是否正确运行：`docker images –filter reference=`*`repositoryUri`*

在输出中，存储库名称从 fluent-bit-multiline-repo 更改为 `repositoryUri`。

d. 通过运行 `aws ecr get-login-password` 命令并指定您要对其进行身份验证的注册表 ID 对 Amazon ECR 进行身份验证：`aws ecr get-login-password | docker login --username AWS --password-stdin registry ID.dkr.ecr.region.amazonaws.com`

   例如：`ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

   此时会显示成功登录消息。

e. 将镜像推送到 Amazon ECR：`docker push registry ID.dkr.ecr.region.amazonaws.com/repository name`

   例如：`docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

为演示多行应用程序生成并上传镜像

此镜像将包括运行应用程序的 Python 脚本文件和示例日志文件。

1. 创建名为 `multiline-app` 的文件夹：`mkdir multiline-app`
2. 创建 Python 脚本文件。

   a. 在 `multiline-app` 文件夹中，创建一个文件并将其命名为 `main.py`。

   b. 将以下内容粘贴到该文件中：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
```

```
print(count)
print("app terminated.")
```

c. 保存 main.py 文件。

3. 创建示例日志文件。

a. 在 multiline-app 文件夹中，创建一个文件并将其命名为 test.log。

b. 将以下内容粘贴到该文件中：

```
panic: my panic

goroutine 4 [running]:
panic(0x45cb40, 0x47ad70)
  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8 sp=0xc42003f710
 pc=0x422f7c
main.main.func1(0xc420024120)
  foo.go:6 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0
 sp=0xc42003f7d8 pc=0x44b4d1
created by main.main
  foo.go:5 +0x58

goroutine 1 [chan receive]:
runtime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00
 pc=0x42503c
runtime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e
runtime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4
runtime.chanrecv1(0xc420024120, 0x0)
  /usr/local/go/src/runtime/chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20
 pc=0x40439b
main.main()
  foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef
runtime.main()
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad
runtime.goexit()
```

```
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc420053fe8
 sp=0xc420053fe0 pc=0x44b4d1

goroutine 2 [force gc (idle)]:
runtime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c
runtime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e
runtime.forcegchelper()
  /usr/local/go/src/runtime/proc.go:238 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8
 pc=0x424e5c
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8
 sp=0xc42003e7e0 pc=0x44b4d1
created by runtime.init.4
  /usr/local/go/src/runtime/proc.go:227 +0x35

goroutine 3 [GC sweep wait]:
runtime.gopark(0x4739b8, 0x4ad7e0, 0x46fdd2, 0xd, 0x419914, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003ef60 sp=0xc42003ef30
 pc=0x42503c
runtime.goparkunlock(0x4ad7e0, 0x46fdd2, 0xd, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003efa0 sp=0xc42003ef60
 pc=0x42512e
runtime.bgsweep(0xc42001e150)
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8
 sp=0xc42003efa0 pc=0x419973
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003efe0
 sp=0xc42003efd8 pc=0x44b4d1
created by runtime.gcenable
  /usr/local/go/src/runtime/mgc.go:216 +0x58
one more line, no multiline
```

c. 保存 `test.log` 文件。

4. 在 `multiline-app` 文件夹中，创建 Dockerfile。

   a. 将以下内容粘贴到该文件中：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log
```

```
RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 保存 Dockerfile 文件。

5. 使用 Dockerfile 生成镜像。

a. 生成镜像：`docker build -t multiline-app-image .`

其中：`multiline-app-image` 是此示例中镜像的名称。

b. 验证是否已正确创建镜像：`docker images –filter reference=multiline-app-image`

如果成功，输出将显示该镜像和 `latest` 标签。

6. 将镜像上载到 Amazon Elastic Container 注册表。

a. 创建用于存储镜像的 Amazon ECR 存储库：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

其中：`multiline-app-repo` 是存储库的名称，`us-east-1` 是此示例中的区域。

输出为您提供了新存储库的详细信息。记下 `repositoryUri` 值，您将需要在后续步骤中使用该值。

b. 使用上一个输出中的 `repositoryUri` 值标记您的镜像：`docker tag` *`multiline-app-image repositoryUri`*

例如：`docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. 运行 Docker 镜像以验证它是否正确运行：`docker images –filter reference=`*`repositoryUri`*

在输出中，存储库名称从 `multiline-app-repo` 更改为 `repositoryUri` 值。

d. 将镜像推送到 Amazon ECR：`docker push` *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository name`*

例如：docker push *xxxxxxxxxxxx*.dkr.ecr.*us-east-1*.amazonaws.com/multiline-app-repo

创建任务定义并运行任务

1. 使用文件名 multiline-task-definition.json 创建任务定义文件。
2. 将以下内容粘贴到 multiline-task-definition.json 文件中：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN,
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-
multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
```

```
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

替换 multiline-task-definition.json 任务定义中的以下内容：

a. *task role ARN*

要查找任务角色 ARN，请转到 IAM 控制台。选择 Roles（角色），然后查找您创建的 ecs-task-role-for-firelens 任务角色。选择角色，然后复制 Summary（摘要）部分中显示的 ARN。

b. *execution role ARN*

要查找执行角色 ARN，请转到 IAM 控制台。选择 Roles（角色），然后查找 ecsTaskExecutionRole 角色。选择角色，然后复制 Summary（摘要）部分中显示的 ARN。

c. *aws_account_id*

要查找您的 aws_account_id，登录 AWS Management Console。在右上角选择您的用户名，然后复制您的账户 ID。

d. *us-east-1*

如有必要，请更换区域。

3. 注册任务定义文件：aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region us-east-1
4. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
5. 在导航窗格中，选择 Task Definitions（任务定义），然后选择 firelens-example-multiline 系列，因为我们已在上面任务定义的第一行中将任务定义注册到这个系列。
6. 选择最新版本。
7. 选择部署、运行任务。
8. 在运行任务页面上，对于集群，选择集群，然后在联网下的子网中，选择任务的可用子网。

9. 选择创建。

验证 Amazon CloudWatch 中的多行日志消息是否已连接

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 从导航窗格中，展开 Logs（日志）并选择 Log groups（日志组）。
3. 选择 multiline-test/applicatio 日志组。
4. 选择日志并查看消息。与解析器文件中的规则匹配的行将串联起来，并显示为单条消息。

   以下日志代码段显示了在单个事件中连接的 Go 堆栈跟踪：

```
{
    "log": "panic: my panic\n\ngoroutine 4 [running]:\npanic(0x45cb40,
 0x47ad70)\n  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8
 sp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n  foo.go:6
 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8
 pc=0x44b4d1\ncreated by main.main\n  foo.go:5 +0x58\n\ngoroutine 1 [chan receive]:
\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n  /usr/
local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c
\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n  /usr/local/go/src/runtime/
chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n
 foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef\nruntime.main()\n
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337
 +0x1 fp=0xc420053fe8 sp=0xc420053fe0 pc=0x44b4d1\n\ngoroutine 2 [force gc
 (idle)]:\nruntime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)\n  /
usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c\nruntime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e\nruntime.forcegchelper()\n  /usr/local/go/src/runtime/proc.go:238
 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8 pc=0x424e5c\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8 sp=0xc42003e7e0
 pc=0x44b4d1\ncreated by runtime.init.4\n  /usr/local/go/src/runtime/proc.go:227
 +0x35\n\ngoroutine 3 [GC sweep wait]:\nruntime.gopark(0x4739b8, 0x4ad7e0,
 0x46fdd2, 0xd, 0x419914, 0x1)\n  /usr/local/go/src/runtime/proc.go:280 +0x12c
 fp=0xc42003ef60 sp=0xc42003ef30 pc=0x42503c\nruntime.goparkunlock(0x4ad7e0,
```

```
 0x46fdd2, 0xd, 0x14, 0x1)\n  /usr/local/go/src/runtime/proc.go:286 +0x5e
 fp=0xc42003efa0 sp=0xc42003ef60 pc=0x42512e\nruntime.bgsweep(0xc42001e150)\n
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8 sp=0xc42003efa0
 pc=0x419973\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1
 fp=0xc42003efe0 sp=0xc42003efd8 pc=0x44b4d1\ncreated by runtime.gcenable\n  /usr/
local/go/src/runtime/mgc.go:216 +0x58",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

以下日志代码段显示了如果您运行的是未配置为连接多行日志消息的 ECS 容器，如何显示相同的事件。日志字段包含单行。

```
{
    "log": "panic: my panic",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
```

> **Note**
>
> 如果您的日志转到日志文件而不是标准输出，我们建议在结尾输入插件中指定 `multiline.parser` 和 `multiline.key_content` 配置参数，而不是筛选条件。

在 Amazon ECS Windows 容器上部署 Fluent Bit

Fluent Bit 是一种快速灵活的日志处理器和路由器，受各种操作系统支持。它可以用来将日志路由到不同的 AWS 目的地，例如 Amazon CloudWatch Logs、Firehose Amazon S3 和 Amazon OpenSearch Service。Fluent Bit 支持常见的合作伙伴解决方案，例如 Datadog、Splunk 和自定义 HTTP 服务器。有关 Fluent Bit 的更多信息，请参阅 Fluent Bit 网站。

为了获得高可用性，大多数区域的 Amazon ECR 公共图库和 Amazon ECR 存储库上的 Amazon ECR 都提供了 AWS for Fluent Bit 映像。有关更多信息，请参阅 GitHub 网站上的 aws-for-fluent-bit 。

本教程介绍如何在 Amazon ECS 中运行的 Windows 实例上部署 Fluent Bit 容器，以便将 Windows 任务生成的日志流式传输到 Amazon CloudWatch 进行集中记录。

本教程使用以下方法：

- Fluent Bit 使用进程守护程序计划策略作为服务运行。此策略可确保 Fluent Bit 的单个实例始终在集群中的容器实例上运行。
  - 使用正向输入插件侦听端口 24224。
  - 向主机公开端口 24224，这样 Docker 运行时就可以使用这个公开的端口向 Fluent Bit 发送日志。
  - 具有允许 Fluent Bit 将日志记录发送到指定目标的配置。
- 使用 fluentd 日志记录驱动程序启动所有其他 Amazon ECS 任务容器。有关更多信息，请参阅 Docker 文档网站上的 Fluentd 日志记录驱动程序。
  - Docker 连接到主机命名空间内本地主机上的 TCP 套接字 24224。
  - Amazon ECS 代理向容器添加标签，其中包括集群名称、任务定义系列名称、任务定义修订版号、任务 ARN 和容器名称。使用 fluentd Docker 日志记录驱动程序的标签选项将相同的信息添加到日志记录中。有关更多信息，请参阅 Docker 文档网站上的标签、labels-regex、env 和 env-regex。
  - 由于 fluentd 日志记录驱动程序的 `async` 选项设置为 `true`，因此当重新启动 Fluent Bit 容器时，Docker 会缓冲日志，直到 Fluent Bit 容器重新启动。您可以通过设置 fluentd-bufffer-limit 选项来增加缓冲区限制。有关更多信息，请参阅 Docker 文档网站上的 fluentd-buffer-limit。

工作流程如下：

- Fluent Bit 容器启动并侦听向主机公开的端口 24224。
- Fluent Bit 使用其任务定义中指定的任务 IAM 角色凭证。
- 在同一实例上启动的其他任务使用 fluentd Docker 日志记录驱动程序连接到端口 24224 上的 Fluent Bit 容器。
- 当应用程序容器生成日志时，Docker 运行时会标记这些记录，添加标签中指定的其他元数据，然后将它们转发到主机命名空间中的端口 24224。
- Fluent Bit 在端口 24224 上接收日志记录，因为它向主机命名空间公开。
- Fluent Bit 执行其内部处理并按照指定路由日志。

本教程使用默认 CloudWatch Fluent Bit 配置，该配置执行以下操作：

- 为每个集群和任务定义系列创建新的日志组。
- 每当启动新任务时，都会为上面生成的日志组中的每个任务容器创建新的日志流。每个流都将标有容器所属的任务 ID。
- 在每个日志条目中添加其他元数据，包括集群名称、任务 ARN、任务容器名称、任务定义系列和任务定义修订版号。

  例如，如果 `task_1` 包含 `container_1` 和 `container_2`，`ask_2` 包含 `container_3`，则以下是 CloudWatch 日志流：

  - `/aws/ecs/windows.ecs_task_1`

    `task-out.TASK_ID.container_1`

    `task-out.TASK_ID.container_2`
  - `/aws/ecs/windows.ecs_task_2`

    `task-out.TASK_ID.container_3`

步骤

- 先决条件
- 步骤 1：创建 IAM 访问角色
- 步骤 2：创建 Amazon ECS Windows 容器实例
- 步骤 3：配置 Fluent Bit
- 步骤 4：注册 Windows Fluent Bit 任务定义，该定义将日志路由到 CloudWatch
- 步骤 5：使用进程守护程序计划策略将 ecs-windows-fluent-bit 任务定义作为 Amazon ECS 服务运行
- 步骤 6：注册生成日志的 Windows 任务定义
- 步骤 7：运行 windows-app-task 任务定义
- 步骤 8：验证 CloudWatch 上的日志
- 步骤 9：清除

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 Command Line Interface。
- `aws-for-fluent-bit` 容器映像可用于以下 Windows 操作系统：
  - Windows Server 2019 Core
  - Windows Server 2019 Full
  - Windows Server 2022 Core
  - Windows Server 2022 Full
- 设置以使用 Amazon ECS 中的步骤已完成。
- 您有一个集群。在本教程中，集群名称为 FluentBit-cluster。
- 您有一个带有公有子网的 VPC，将在该子网中启动 EC2 实例。您可以使用您的原定设置 VPC。您也可以使用允许 Amazon CloudWatch 端点到达子网的私有子网。有关 Amazon CloudWatch 端点的更多信息，请参阅 AWS 一般参考 中的 Amazon CloudWatch 端点和配额。有关如何使用 Amazon VPC 向导创建 VPC 的信息，请参阅 the section called “创建 Virtual Private Cloud”。

步骤 1：创建 IAM 访问角色

创建 Amazon ECS IAM 角色。

1. 创建名为“ecsInstanceRol”的 Amazon ECS 容器实例角色。有关更多信息，请参阅 Amazon ECS 容器实例 IAM 角色。
2. 为名为 `fluentTaskRole` 的 Fluent Bit 任务创建一个 IAM 角色。有关更多信息，请参阅 the section called “任务 IAM 角色”。

   在此 IAM 角色中被授予的 IAM 权限由任务容器承担。为了允许 Fluent Bit 向 CloudWatch 发送日志，您需要为任务 IAM 角色附加以下权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents"
        ],
        "Resource": "*"
```

```
    }
    ]
}
```

3. 将 策略附加到该角色。

   a. 将上述内容保存在名为 `fluent-bit-policy.json` 的文件中。

   b. 运行以下命令将内联策略附加到 `fluentTaskRole` IAM 角色。

```
aws iam put-role-policy --role-name fluentTaskRole --policy-name
 fluentTaskPolicy --policy-document file://fluent-bit-policy.json
```

步骤 2：创建 Amazon ECS Windows 容器实例

创建 Amazon ECS Windows 容器实例。

创建 Amazon ECS 实例

1. 使用 `aws ssm get-parameters` 命令检索托管您 VPC 的区域的 AMI ID。有关更多信息，请参阅检索经 Amazon ECS 优化的 AMI 元数据。
2. 用来创建启动实例的 Amazon EC2 控制台。

   a. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

   b. 从导航栏中，选择要使用的区域。

   c. 从 EC2 控制面板中，选择 Launch Instance（启动实例）。

   d. 对于 Name (名称)，输入唯一的名称。

   e. 对于 Application and OS Images (Amazon Machine Image)（应用程序和操作系统映像 [Amazon 系统映像]），选择您在第一步中检索的 AMI。

   f. 对于实例类型，选择 `t3.xlarge`。

   g. 对于 Key pair (login)（密钥对 [登录]），选择一个密钥对。

   h. 在 Network settings（网络设置）下，对于 Security group（安全组），可以选择现有安全组或创建新安全组。

   i. 在 Network settings（网络设置）下，对于 Auto-assign Public IP（自动分配公有 IP），选择 Enable（启用）。

   j. 在 Advanced details（高级详细信息）下，对于 IAM instance profile（IAM 实例配置文件），选择 ecsInstanceRole。

k. 使用以下用户数据配置您的 Amazon ECS 容器实例。在 Advanced Details（高级详细信息）下，将以下脚本粘贴到 User data（用户数据）字段中，将 *cluster_name* 替换为您的集群的名称。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster cluster-name -EnableTaskENI -EnableTaskIAMRole -
LoggingDrivers '["awslogs","fluentd"]'
</powershell>
```

l. 准备好后，选中确认字段，然后选择 Launch Instances。

m. 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。

步骤 3：配置 Fluent Bit

您可以使用 AWS 提供的以下默认配置快速入门：

- Amazon CloudWatch 是基于《Fluent Bit 官方手册》上的适用于 Amazon CloudWatch 的 Fluent Bit 插件。

或者，您可以使用 AWS 提供的其他默认配置。有关更多信息，请参阅 Github 网站上关于 `aws-for-fluent-bit` 的覆盖 Windows 映像的入口点。

Amazon CloudWatch Fluent Bit 的默认配置如下所示。

替换以下变量：

- *region*，用您要将 Amazon CloudWatch Logs 发送到的区域替换。

```
[SERVICE]
    Flush               5
    Log_Level           info
    Daemon              off

[INPUT]
    Name                forward
    Listen              0.0.0.0
    Port                24224
```

```
    Buffer_Chunk_Size   1M
    Buffer_Max_Size     6M
    Tag_Prefix          ecs.

# Amazon ECS agent adds the following log keys as labels to the docker container.
# We would use fluentd logging driver to add these to log record while sending it to
 Fluent Bit.
[FILTER]
    Name                modify
    Match               ecs.*
    Rename              com.amazonaws.ecs.cluster ecs_cluster
    Rename              com.amazonaws.ecs.container-name ecs_container_name
    Rename              com.amazonaws.ecs.task-arn ecs_task_arn
    Rename              com.amazonaws.ecs.task-definition-family
 ecs_task_definition_family
    Rename              com.amazonaws.ecs.task-definition-version
 ecs_task_definition_version

[FILTER]
    Name                rewrite_tag
    Match               ecs.*
    Rule                $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9]+)$
 out.$3.$ecs_container_name false
    Emitter_Name        re_emitted

[OUTPUT]
    Name                cloudwatch_logs
    Match               out.*
    region              region
    log_group_name      fallback-group
    log_group_template  /aws/ecs/$ecs_cluster.$ecs_task_definition_family
    log_stream_prefix   task-
    auto_create_group   On
```

进入 Fluent Bit 的每个日志都有您指定的标签，或者在您不提供标签时自动生成。标签可用于将不同的日志路由到不同的目标。有关更多信息，请参阅《Fluent Bit 官方手册》中的标签。

上面描述的 Fluent Bit 配置具有以下属性：

- 正向输入插件侦听 TCP 端口 24224 上的传入流量。
- 在该端口上收到的每个日志条目都有一个标签，正向输入插件会修改该标签，以将 `ecs.` 字符串作为记录的前缀。

- Fluent Bit 内部管道使用 Match 正则表达式路由日志条目以修改筛选条件。此筛选条件将日志记录 JSON 中的密钥替换为 Fluent Bit 可以使用的格式。
- 然后，rewrite_tag 筛选条件会使用修改后的日志条目。此筛选条件将日志记录的标签更改为格式 out.*TASK_ID*.*CONTAINER_NAME*。
- 新标签将路由到输出 cloudwatch_logs 插件，该插件使用 CloudWatch 输出插件的 `log_group_template` 和 `log_stream_prefix` 选项创建如前所述的日志组和流。有关更多信息，请参阅《Fluent Bit 官方手册》中的配置参数。

步骤 4：注册 Windows Fluent Bit 任务定义，该定义将日志路由到 CloudWatch

注册 Windows Fluent Bit 任务定义，该定义将日志路由到 CloudWatch。

> **Note**
>
> 此任务定义将 Fluent Bit 容器端口 24224 向主机端口 24224 公开。确认此端口未在您的 EC2 实例安全组中打开，以防止外部访问。

注册任务定义

1. 使用以下内容创建名为 `fluent-bit.json` 的文件。

   替换以下变量：

   - 用您的任务 IAM 角色的 Amazon 资源名称（ARN）替换 *task-iam-role*
   - 用您的任务运行所在的区域替换 *region*

```
{
  "family": "ecs-windows-fluent-bit",
  "taskRoleArn": "task-iam-role",
  "containerDefinitions": [
    {
      "name": "fluent-bit",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-
bit:windowsservercore-latest",
      "cpu": 512,
      "portMappings": [
```

```
        {
          "hostPort": 24224,
          "containerPort": 24224,
          "protocol": "tcp"
        }
      ],
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "C:\\entrypoint.ps1 -ConfigFile C:\\ecs_windows_forward_daemon\
\cloudwatch.conf"
      ],
      "environment": [
        {
          "name": "AWS_REGION",
          "value": "region"
        }
      ],
      "memory": 512,
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fluent-bit-logs",
          "awslogs-region": "region",
          "awslogs-stream-prefix": "flb",
          "awslogs-create-group": "true"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 运行以下命令注册任务定义。

```
aws ecs register-task-definition --cli-input-json file://fluent-bit.json --
region region
```

您可以通过运行 `list-task-definitions` 命令列出您的账户的任务定义。输出显示了可以与 `run-task` 或 `start-task` 一起使用的系列和修订版本值。

步骤 5：使用进程守护程序计划策略将 `ecs-windows-fluent-bit` 任务定义作为 Amazon ECS 服务运行

为您的账户注册任务定义后，您可以在集群中运行任务。在本教程中，您将在 `FluentBit-cluster` 集群中运行 `ecs-windows-fluent-bit:1` 任务定义的一个实例。在使用进程守护程序计划策略的服务中运行任务，该策略可确保 Fluent Bit 的单个实例始终在每个容器实例上运行。

运行任务

1. 运行以下命令将 `ecs-windows-fluent-bit:1` 任务定义（在上一步中注册）作为服务启动。

   > **Note**
   >
   > 此任务定义使用 `awslogs` 日志记录驱动程序，您的容器实例需要具有必要的权限。

   替换以下变量：

   - 用您的服务运行所在的区域替换 *region*

   ```
   aws ecs create-service \
       --cluster FluentBit-cluster \
       --service-name FluentBitForwardDaemonService \
       --task-definition ecs-windows-fluent-bit:1 \
       --launch-type EC2 \
       --scheduling-strategy DAEMON \
       --region region
   ```

2. 运行以下命令列出您的任务。

   替换以下变量：

   - 用您的服务任务运行所在的区域替换 *region*

```
aws ecs list-tasks --cluster FluentBit-cluster --region region
```

步骤 6：注册生成日志的 Windows 任务定义

注册生成日志的任务定义。此任务定义部署 Windows 容器映像，该映像将每秒向 stdout 写入一个增量数字。

任务定义使用 fluentd 日志记录驱动程序，该驱动程序连接到 Fluent Bit 插件侦听的端口 24224。Amazon ECS 代理使用标签标记每个 Amazon ECS 容器，包括集群名称、任务 ARN、任务定义系列名称、任务定义修订版号和任务容器名称。这些键值标签将传递给 Fluent Bit。

> **Note**
>
> 此任务使用 default 网络模式。但是，您还可以使用 awsvpc 网络模式运行任务。

注册任务定义

1. 使用以下内容创建名为 windows-app-task.json 的文件。

```
{
  "family": "windows-app-task",
  "containerDefinitions": [
    {
      "name": "sample-container",
      "image": "mcr.microsoft.com/windows/servercore:ltsc2019",
      "cpu": 512,
      "memory": 512,
      "essential": true,
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "$count=1;while(1) { Write-Host $count; sleep 1; $count=$count+1;}"
      ],
      "logConfiguration": {
        "logDriver": "fluentd",
```

```
        "options": {
          "fluentd-address": "localhost:24224",
          "tag": "{{ index .ContainerLabels \"com.amazonaws.ecs.task-definition-
family\" }}",
          "fluentd-async": "true",
          "labels": "com.amazonaws.ecs.cluster,com.amazonaws.ecs.container-
name,com.amazonaws.ecs.task-arn,com.amazonaws.ecs.task-definition-
family,com.amazonaws.ecs.task-definition-version"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 运行以下命令注册任务定义。

   替换以下变量：

   - 用您的任务运行所在的区域替换 *region*

```
aws ecs register-task-definition --cli-input-json file://windows-app-task.json --
region region
```

   您可以通过运行 `list-task-definitions` 命令列出您的账户的任务定义。输出显示了可以与 `run-task` 或 `start-task` 一起使用的系列和修订版本值。

步骤 7：运行 windows-app-task 任务定义

注册 `windows-app-task` 任务定义后，在 `FluentBit-cluster` 集群中运行。

运行任务

1. 运行您在上一步中注册的 `windows-app-task:1` 任务定义。

   替换以下变量：

   - 用您的任务运行所在的区域替换 *region*

```
aws ecs run-task --cluster FluentBit-cluster --task-definition windows-app-task:1
 --count 2 --region region
```

2. 运行以下命令列出您的任务。

```
aws ecs list-tasks --cluster FluentBit-cluster
```

步骤 8：验证 CloudWatch 上的日志

要验证您的 Fluent Bit 设置，请在 CloudWatch 控制台中检查以下日志组：

- `/ecs/fluent-bit-logs` — 这是与在容器实例上运行的 Fluent Bit 进程守护程序容器相对应的日志组。
- `/aws/ecs/FluentBit-cluster.windows-app-task` — 这是与 `FluentBit-cluster` 集群中为 `windows-app-task` 任务定义系列启动的所有任务相对应的日志组。

  `task-out.FIRST_TASK_ID.sample-container` — 此日志流包含示例容器任务容器中任务的第一个实例生成的所有日志。

  `task-out.SECOND_TASK_ID.sample-container` — 此日志流包含示例容器任务容器中任务的第二个实例生成的所有日志。

`task-out.TASK_ID.sample-container` 日志流具有类似于以下内容的字段：

```
{
    "source": "stdout",
    "ecs_task_arn": "arn:aws:ecs:region:0123456789012:task/FluentBit-
cluster/13EXAMPLE",
    "container_name": "/ecs-windows-app-task-1-sample-container-cEXAMPLE",
    "ecs_cluster": "FluentBit-cluster",
    "ecs_container_name": "sample-container",
    "ecs_task_definition_version": "1",
    "container_id": "61f5e6EXAMPLE",
    "log": "10",
    "ecs_task_definition_family": "windows-app-task"
}
```

验证 Fluent Bit 设置

1. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Log groups（日志组）。确保您位于将 Fluent Bit 部署到您的容器的区域中。

   在 AWS 区域 的日志组列表中，您将会看到以下内容：

   - `/ecs/fluent-bit-logs`
   - `/aws/ecs/FluentBit-cluster.windows-app-task`

   如果看到这些日志组，则代表 Fluent Bit 设置已验证。

步骤 9：清除

完成本教程后，请清除与本教程关联的资源，以避免对您未使用的资源产生费用。

清除教程资源

1. 停止 `windows-simple-task` 任务和 `ecs-fluent-bit` 任务。有关更多信息，请参阅 the section called “停止任务”。
2. 运行以下命令删除 `/ecs/fluent-bit-logs` 日志组。有关删除日志组的更多信息，请参阅《AWS Command Line Interface 参考》中的 delete-log-group。

   ```
   aws logs delete-log-group --log-group-name /ecs/fluent-bit-logs
   aws logs delete-log-group --log-group-name /aws/ecs/FluentBit-cluster.windows-app-task
   ```

3. 运行以下命令终止实例。

   ```
   aws ec2 terminate-instances --instance-ids instance-id
   ```

4. 运行以下命令删除 IAM 角色。

   ```
   aws iam delete-role --role-name ecsInstanceRole
   aws iam delete-role --role-name fluentTaskRole
   ```

5. 运行以下命令删除 Amazon ECS 集群。

```
aws ecs delete-cluster --cluster FluentBit-cluster
```

将 gMSA 用于 Amazon ECS 上的 EC2 Linux 容器

Amazon ECS 通过称为组托管服务账户（gMSA）的特殊服务账户支持 EC2 上的 Linux 容器的 Active Directory 身份验证。

基于 Linux 的网络应用程序（例如 .NET Core 应用程序）可以使用 Active Directory 来促进用户和服务之间的身份验证和授权管理。通过设计与 Active Directory 集成并在加入域的服务器上运行的应用程序，您可以使用此功能。但是，由于 Linux 容器无法加入域，因此您需要配置一个 Linux 容器来使用 gMSA 运行。

使用 gMSA 运行的 Linux 容器依赖于在容器主机 Amazon EC2 实例上运行的 `credentials-fetcher` 进程守护程序。也就是说，进程守护程序从 Active Directory 域控制器检索 gMSA 凭证，然后将这些凭证传输到容器实例。有关服务账户的更多信息，请参阅 Microsoft Learn 网站上的为 Windows 容器创建 gMSAs。

注意事项

在将 gMSA 用于 Linux 容器之前，请注意以下各项：

- 如果您的容器在 EC2 上运行，则可以将 gMSA 用于 Windows 容器和 Linux 容器。有关如何在 Fargate 上使用 gMSA Linux 容器的信息，请参阅 将 gMSA 用于 Fargate 上的 Linux 容器。
- 您可能需要一台已加入域的 Windows 计算机才能完成先决条件。例如，您可能需要一台已加入域的 Windows 计算机才能使用 PowerShell 在 Active Directory 中创建 gMSA。RSAT Active Director PowerShell 工具仅适用于 Windows。有关更多信息，请参阅安装 Active Directory 管理工具。
- 您可以在无域 gMSA 和将每个实例加入单个域之间进行选择。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

  然后，选择数据存储用于 CredSpec，以及无域 gMSA 的 Active Directory 用户凭证（可选）。

  Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到容器的 gMSA 元数据。您可以生成 CredSpec 文件，然后将其存储在下表中的一个 CredSpec 存储选项中，该存储选项特定于容器实例的操作系统。要使用无域方法，CredSpec 文件中的可选部分可以在下表中的其中一个 domainless user credentials 存储选项中指定凭证，这些凭证特定于容器实例的操作系统。

按操作系统划分的 gMSA 数据存储选项

存储位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	无域用户凭证	无域用户凭证
Amazon EC2 Systems Manager Parameter Store	CredSpec	CredSpec，无域用户凭证
本地文件	不适用	CredSpec

先决条件

在 Amazon ECS 上为 Linux 容器使用 gMSA 之前，请确保完成以下操作：

- 您可以使用您希望容器访问的资源设置 Active Directory 域。Amazon ECS 支持以下设置：
  - AWS Directory Service Active Directory。AWS Directory Service 是托管在 Amazon EC2 上的 AWS 托管式 Active Directory。有关更多信息，请参阅 AWS Directory Service 管理指南中的 AWS 托管 Microsoft AD 入门。
  - 本地 Active Directory。您必须确保 Amazon ECS Linux 容器实例可以加入域。有关更多信息，请参阅 AWS Direct Connect。
- 您在 Active Directory 中有现有的 gMSA 账户。有关更多信息，请参阅 将 gMSA 用于 Amazon ECS 上的 EC2 Linux 容器。
- 您已在 Amazon ECS Linux 容器实例上安装并正在运行 `credentials-fetcher` 进程守护程序。您还向 `credentials-fetcher` 进程守护程序添加了一组初始凭证，以便通过 Active Directory 进行身份验证。

> **Note**
>
> `credentials-fetcher` 进程守护程序仅适用于 Amazon Linux 2023 和 Fedora 37 及更高版本。该进程守护程序不适用于 Amazon Linux 2。有关更多信息，请参阅 GitHub 上的 aws/credentials-fetcher。

- 您可以为 `credentials-fetcher` 进程守护程序设置凭证，以便通过 Active Directory 进行身份验证。凭证必须是有权访问 gMSA 账户的 Active Directory 安全组的成员。决定是要将实例加入域，还是要使用无域 gMSA。 里面有多个选项。
- 您已添加所需的 IAM 权限。所需的权限取决于您为初始凭证和存储凭证规范选择的方法：
  - 如果您使用无域 gMSA 作为初始凭证，则任务执行角色需要 AWS Secrets Manager 的 IAM 权限。
  - 如果您将凭证规范存储在 SSM Parameter Store 中，则任务执行角色需要获得 Amazon EC2 Systems Manager Parameter Store 的 IAM 权限。
  - 如果您将凭证规范存储在 Amazon S3 中，则任务执行角色需要 Amazon Simple Storage Service 的 IAM 权限。

在 Amazon ECS 上设置支持 gMSA 的 Linux 容器

准备基础设施

以下步骤是仅执行一次的注意事项和设置。完成这些步骤后，您可以自动创建容器实例，以重复使用此配置。

决定如何提供初始凭证，并在可重复使用的 EC2 启动模板中配置 EC2 用户数据以安装 `credentials-fetcher` 进程守护程序。

1. 决定是要将实例加入域，还是要使用无域 gMSA。
   - 将 EC2 实例加入 Active Directory 域
     - 按用户数据加入实例

       在 EC2 启动模板中，将加入 Active Directory 域的步骤添加到 EC2 用户数据中。多个 Amazon EC2 Auto Scaling 组可以使用同一个启动模板。

       您可以使用 Fedora 文档中的这些步骤加入 Active Directory 或 FreeIPA 域。
   - 为无域 gMSA 创建 Active Directory 用户

     `credentials-fetcher` 进程守护程序有一个称为无域 gMSA 的功能。此功能需要域，但是无需 EC2 实例加入该域。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。相反，您可以在 CredSpec 文件中的 AWS Secrets Manager 中提供密钥的名称。密钥必须包含用户名、密码和要登录到的域。

该功能受支持，并且可以与 Linux 和 Windows 容器结合使用。

此功能与该 gMSA support for non-domain-joined container hosts 功能类似。有关 Windows 功能的更多信息，请参阅 Microsoft Learn 网站上的 gMSA 架构和改进。

a. 在 Active Directory 域中创建用户。Active Directory 中的用户必须具有访问您在任务中使用的 gMSA 服务账户的权限。

b. 在 Active Directory 中创建用户后，请在 AWS Secrets Manager 中创建密钥。有关更多信息，请参阅创建一个 AWS Secrets Manager 密钥。

c. 将用户的用户名、密码和域分别输入名为 `username`、`password` 和 `domainName` 的 JSON 键值对。

```
{"username":"username","password":"passw0rd", "domainName":"example.com"}
```

d. 向 CredSpec 文件中添加服务账户的配置。其他 `HostAccountConfig` 中包含 Secrets Manager 密钥的 Amazon 资源名称（ARN）。

在 Windows 上，`PluginGUID` 必须与以下示例代码段中的 GUID 相匹配。在 Linux 上，`PluginGUID` 会被忽略。将 `MySecret` 示例替换为密钥的 Amazon 资源名称（ARN）。

```
    "ActiveDirectoryConfig": {
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
```

e. 无域 gMSA 功能需要任务执行角色中的额外权限。执行步骤（可选）无域 gMSA 密钥。

2. 配置实例并安装 **`credentials-fetcher`** 进程守护程序

您可以在 EC2 启动模板中使用用户数据脚本安装 `credentials-fetcher` 进程守护程序。以下示例演示了两种类型的用户数据，`cloud-config` YAML 或 bash 脚本。这些示例适用于 Amazon Linux 2023（AL2023）。将 `MyCluster` 替换为要让这些实例加入的 Amazon ECS 集群的名称。

- **`cloud-config`** YAML

```
Content-Type: text/cloud-config
package_reboot_if_required: true
packages:
  # prerequisites
  - dotnet
  - realmd
  - oddjob
  - oddjob-mkhomedir
  - sssd
  - adcli
  - krb5-workstation
  - samba-common-tools
  # https://github.com/aws/credentials-fetcher gMSA credentials management for
 containers
  - credentials-fetcher
write_files:
# configure the ECS Agent to join your cluster.
# replace MyCluster with the name of your cluster.
- path: /etc/ecs/ecs.config
  owner: root:root
  permissions: '0644'
  content: |
    ECS_CLUSTER=MyCluster
    ECS_GMSA_SUPPORTED=true
runcmd:
# start the credentials-fetcher daemon and if it succeeded, make it start after
 every reboot
- "systemctl start credentials-fetcher"
- "systemctl is-active credentials-fetch && systemctl enable credentials-
fetcher"
```

- bash 脚本

  如果您更熟悉 bash 脚本并且有多个变量要写入 `/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。

```
#!/usr/bin/env bash
set -euxo pipefail

# prerequisites
```

```
timeout 30 dnf install -y dotnet realmd oddjob oddjob-mkhomedir sssd adcli
 krb5-workstation samba-common-tools
# install https://github.com/aws/credentials-fetcher gMSA credentials
 management for containers
timeout 30 dnf install -y credentials-fetcher

# start credentials-fetcher
systemctl start credentials-fetcher
systemctl is-active credentials-fetch && systemctl enable credentials-fetcher

cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_GMSA_SUPPORTED=true
EOF
```

您可以在 /etc/ecs/ecs.config 中设置的 credentials-fetcher 进程守护程序有一些可选的配置变量。建议您在 YAML 块中或类似于前面的示例的 heredoc 中设置用户数据中的变量。这样，在多次编辑文件时，可以防止可能出现的部分配置问题。有关 ECS 代理配置的更多信息，请参阅 GitHub 上的 Amazon ECS 容器代理。

- 或者，如果您更改 credentials-fetcher 进程守护程序配置以将套接字移到其他位置，则可以使用变量 CREDENTIALS_FETCHER_HOST。

设置权限和密钥

针对每个应用程序和每个任务定义，执行一次以下步骤。建议您使用以下最佳实践：授予最低权限和限制策略中使用的权限。这样，每个任务只能读取它需要的密钥。

1. （可选）无域 gMSA 密钥

   如果您使用实例未加入域的无域方法，请按照此步骤操作。

   您还必须将以下权限作为内联策略添加到任务执行 IAM 角色。这样做可以让 credentials-fetcher 进程守护程序访问 Secrets Manager 密钥。将 MySecret 示例替换为 Resource 列表中的密钥的 Amazon 资源名称（ARN）。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:secret:MySecret"
            ]
        }
    ]
}
```

> **Note**
>
> 如果您使用自己的 KMS 密钥来加密您的密钥，则必须向该角色添加必要的权限并将此角色添加到 AWS KMS 密钥政策中。

2. 决定是使用 SSM Parameter Store 还是 S3 来存储 CredSpec

Amazon ECS 支持使用以下方法在任务定义的 `credentialSpecs` 字段中引用文件路径。

如果您将实例加入单个域，请使用字符串中 ARN 开头的前缀 `credentialspec:`。如果您使用无域 gMSA，则使用 `credentialspecdomainless:`。

有关 CredSpec 的更多信息，请参阅 凭证规范文件。

- Amazon S3 存储桶

  将凭证规范添加到 Amazon S3 存储桶。然后在任务定义的 `credentialSpecs` 字段中引用 Amazon S3 存储桶的 Amazon 资源名称（ARN）。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/
${ObjectName}"
            ],
            ...
        }
```

```
    ],
    ...
}
```

为了让您的任务可以访问 S3 存储桶，请将以下权限作为内联策略添加到 Amazon ECS 任务执行 IAM 角色中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/{object}"
            ]
        }
    ]
}
```

- SSM Parameter Store 参数

  将凭证规范添加到 SSM Parameter Store 参数中。然后在任务定义的 `credentialSpecs` 字段中引用 SSM Parameter Store 参数的 Amazon 资源名称（ARN）。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:ssm:aws-
region:111122223333:parameter/parameter_name"
            ],
            ...
```

```
        }
    ],
    ...
}
```

为了让您的任务可以访问 SSM Parameter Store 参数，请将以下权限作为内联策略添加到 Amazon ECS 任务执行 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

凭证规范文件

Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到 Linux 容器的 gMSA 元数据。您可以生成 CredSpec 并在任务定义的 `credentialSpecs` 字段中引用该文件。CredSpec 文件不包含任何机密。

下面是一个 CredSpec 示例文件。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
```

```
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

创建CredSpec

您可以通过在已加入域的 Windows 计算机上使用 CredSpec PowerShell 模块来创建 CredSpec。按照 Microsoft Learn 网站上的创建凭证规范中的步骤进行操作。

将 gMSA 用于 Fargate 上的 Linux 容器

Amazon ECS 通过称为组托管服务账户（gMSA）的特殊服务账户支持 Fargate 上的 Linux 容器的 Active Directory 身份验证。

基于 Linux 的网络应用程序（例如 .NET Core 应用程序）可以使用 Active Directory 来促进用户和服务之间的身份验证和授权管理。通过设计与 Active Directory 集成并在加入域的服务器上运行的应用程序，您可以使用此功能。但是，由于 Linux 容器无法加入域，因此您需要配置一个 Linux 容器来使用 gMSA 运行。

注意事项

在将 gMSA 用于 Fargate 上的 Linux 容器之前，请注意以下各项：

- 您必须运行平台版本 1.4 或更高版本。

- 您可能需要一台已加入域的 Windows 计算机才能完成先决条件。例如，您可能需要一台已加入域的 Windows 计算机才能使用 PowerShell 在 Active Directory 中创建 gMSA。RSAT Active Director PowerShell 工具仅适用于 Windows。有关更多信息，请参阅安装 Active Directory 管理工具。
- 您必须使用无域 gMSA。

  Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到容器的 gMSA 元数据。您将生成 CredSpec 文件，然后将其存储在 Amazon S3 存储桶中。
- 一个任务只能支持一个 Active Directory。

先决条件

在 Amazon ECS 上为 Linux 容器使用 gMSA 之前，请确保完成以下操作：

- 您可以使用您希望容器访问的资源设置 Active Directory 域。Amazon ECS 支持以下设置：
  - AWS Directory Service Active Directory。AWS Directory Service 是托管在 Amazon EC2 上的 AWS 托管式 Active Directory。有关更多信息，请参阅 AWS Directory Service 管理指南中的 AWS 托管 Microsoft AD 入门。
  - 本地 Active Directory。您必须确保 Amazon ECS Linux 容器实例可以加入域。有关更多信息，请参阅 AWS Direct Connect。
- 您在 Active Directory 中有一个现有 gMSA 账户，以及一个有权访问 gMSA 服务账户的用户。有关更多信息，请参阅 为无域 gMSA 创建 Active Directory 用户。
- 您拥有 Amazon S3 存储桶。有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。

在 Amazon ECS 上设置支持 gMSA 的 Linux 容器

准备基础设施

以下步骤是仅执行一次的注意事项和设置。

- 为无域 gMSA 创建 Active Directory 用户

  当您使用无域 gMSA 时，该容器未加入该域。在容器上运行的其他应用程序无法使用凭证访问该域。使用不同域的任务可以在同一个容器上运行。您可以在 CredSpec 文件中的 AWS Secrets Manager 中提供密钥的名称。密钥必须包含用户名、密码和要登录到的域。

此功能与该 gMSA support for non-domain-joined container hosts 功能类似。有关 Windows 功能的更多信息，请参阅 Microsoft Learn 网站上的 gMSA 架构和改进。

a. 在 Active Directory 域中配置用户。Active Directory 中的用户必须具有访问您在任务中使用的 gMSA 服务账户的权限。
b. 您有一个 VPC 和可以解析 Active Directory 域名的子网。使用指向 Active Directory 服务名称的域名配置具有 DHCP 选项的 VPC。有关如何为 VPC 配置 DHCP 选项的信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的使用 DHCP 选项集。
c. 在 AWS Secrets Manager 中创建密钥。
d. 创建凭证规范文件。

设置权限和密钥

针对每个应用程序和每个任务定义，执行一次以下步骤。建议您使用以下最佳实践：授予最低权限和限制策略中使用的权限。这样，每个任务只能读取它需要的密钥。

1. 在 Active Directory 域中创建用户。Active Directory 中的用户必须具有访问您在任务中使用的 gMSA 服务账户的权限。
2. 创建 Active Directory 用户后，请在 AWS Secrets Manager 中创建密钥。有关更多信息，请参阅创建一个 AWS Secrets Manager 密钥。
3. 将用户的用户名、密码和域分别输入名为 `username`、`password` 和 `domainName` 的 JSON 键值对。

```
{"username":"username","password":"passw0rd", "domainName":"example.com"}
```

4. 您还必须将以下权限作为内联策略添加到任务执行 IAM 角色。这样做可以让 `credentials-fetcher` 进程守护程序访问 Secrets Manager 密钥。将 `MySecret` 示例替换为 `Resource` 列表中的密钥的 Amazon 资源名称（ARN）。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
```

```
        "arn:aws:secretsmanager:aws-region:111122223333:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您使用自己的 KMS 密钥来加密您的密钥，则必须向该角色添加必要的权限并将此角色添加到 AWS KMS 密钥政策中。

5. 将凭证规范添加到 Amazon S3 存储桶。然后在任务定义的 `credentialSpecs` 字段中引用 Amazon S3 存储桶的 Amazon 资源名称（ARN）。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

为了让您的任务可以访问 S3 存储桶，请将以下权限作为内联策略添加到 Amazon ECS 任务执行 IAM 角色中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:ListObject"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

凭证规范文件

Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到 Linux 容器的 gMSA 元数据。您可以生成 CredSpec 并在任务定义的 `credentialSpecs` 字段中引用该文件。CredSpec 文件不包含任何机密。

下面是一个 CredSpec 示例文件。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
```

```
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

创建 CredSpec 并将其上传到 Amazon S3

您可以通过在已加入域的 Windows 计算机上使用 CredSpec PowerShell 模块来创建 CredSpec。按照 Microsoft Learn 网站上的创建凭证规范中的步骤进行操作。

创建凭证规范文件后，将其上传到 Amazon S3 存储桶。将 CredSpec 文件复制到您正在其中运行 AWS CLI 命令的计算机或环境中。

运行以下 AWS CLI 命令以将 CredSpec 上载到 Amazon S3。将 `MyBucket` 替换为您的 Amazon S3 存储桶的名称。您可以将文件作为对象存储在任何存储桶和位置，但您必须允许访问该存储桶和您附加到任务执行角色的策略中的位置。

对于 PowerShell，使用以下命令：

```
$ Write-S3Object -BucketName "MyBucket" -Key "ecs-domainless-gmsa-credspec" -File
 "gmsa-cred-spec.json"
```

以下 AWS CLI 命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。

```
$ aws s3 cp gmsa-cred-spec.json \
s3://MyBucket/ecs-domainless-gmsa-credspec
```

通过 AWS CLI 将 Amazon ECS Windows 容器与无域 gMSA 结合使用

以下教程演示如何创建一个 Amazon ECS 任务，该任务运行具有凭证的 Windows 容器，以使用 AWS CLI 访问 Active Directory。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

主题

- 先决条件

- 步骤 1：在 Active Directory 域服务（AD DS）上创建和配置 gMSA 账户
- 步骤 2：将凭证上传到 Secrets Manager
- 步骤 3：修改您的 CredSpec JSON 以包含无域 gMSA 信息
- 步骤 4：将 CredSpec 上传到 Amazon S3
- 步骤 5：（可选）创建 Amazon ECS 集群
- 步骤 6：为容器实例创建 IAM 角色
- 步骤 7：创建自定义任务执行角色
- 步骤 8：为 Amazon ECS Exec 创建任务角色
- 步骤 9：注册一个使用无域 gMSA 的任务定义
- 步骤 10：将 Windows 容器实例注册到集群
- 步骤 11：验证容器实例
- 步骤 12：运行 Windows 任务
- 步骤 13：验证容器是否有 gMSA 凭证
- 步骤 14：清除
- 调试适用于 Windows 容器的 Amazon ECS 无域 gMSA

先决条件

本教程假设以下先决条件已完成：

- 设置以使用 Amazon ECS 中的步骤已完成。
- 您的 AWS 用户具有 AmazonECS_FullAccess IAM policy 示例中指定的所需权限。
- 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。
- 您可以使用您希望容器访问的资源设置 Active Directory 域。Amazon ECS 支持以下设置：
  - AWS Directory Service Active Directory。AWS Directory Service 是托管在 Amazon EC2 上的 AWS 托管式 Active Directory。有关更多信息，请参阅 AWS Directory Service 管理指南中的 AWS 托管 Microsoft AD 入门。
  - 本地 Active Directory。您必须确保 Amazon ECS Linux 容器实例可以加入域。有关更多信息，请参阅 AWS Direct Connect。
- 您有一个 VPC 和可以解析 Active Directory 域名的子网。

- 您可以在无域 gMSA 和将每个实例加入单个域之间进行选择。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

  然后，选择数据存储用于 CredSpec，以及无域 gMSA 的 Active Directory 用户凭证（可选）。

  Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到容器的 gMSA 元数据。您可以生成 CredSpec 文件，然后将其存储在下表中的一个 CredSpec 存储选项中，该存储选项特定于容器实例的操作系统。要使用无域方法，CredSpec 文件中的可选部分可以在下表中的其中一个 domainless user credentials 存储选项中指定凭证，这些凭证特定于容器实例的操作系统。

  按操作系统划分的 gMSA 数据存储选项

存储位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	无域用户凭证	无域用户凭证
Amazon EC2 Systems Manager Parameter Store	CredSpec	CredSpec，无域用户凭证
本地文件	不适用	CredSpec

- （可选）AWS CloudShell 是一种为客户提供命令行的工具，而无需创建自己的 EC2 实例。有关更多信息，请参阅AWS CloudShell《用户指南》中的什么是AWS CloudShell？。

步骤 1：在 Active Directory 域服务（AD DS）上创建和配置 gMSA 账户

在 Active Directory 域上创建和配置 gMSA 账户。

1. 生成密钥分发服务根密钥

> **Note**
>
> 如果您使用 AWS Directory Service，可以跳过此步骤。

KDS 根密钥和 gMSA 权限是使用您的 AWS 托管的 Microsoft AD 配置的。

如果您尚未在域中创建 gMSA 服务账户，则需要先生成密钥分发服务（KDS）根密钥。KDS 负责创建、轮换 gMSA 密码并将其发布给授权的主机。当 `ccg.exe` 需要检索 gMSA 凭证时，它会联系 KDS 以检索当前密码。

要检查是否已创建 KDS 根密钥，请使用 `ActiveDirectory` PowerShell 模块在域控制器上以域管理员权限运行以下 PowerShell cmdlet。有关该模块的更多信息，请参阅 Microsoft Learn 网站上的 ActiveDirectory 模块。

```
PS C:\> Get-KdsRootKey
```

如果命令返回密钥 ID，您可以跳过此步骤的其余部分。否则，请通过运行以下命令创建 KDS 根密钥：

```
PS C:\> Add-KdsRootKey -EffectiveImmediately
```

尽管该命令的参数 `EffectiveImmediately` 暗示密钥立即生效，但您需要等待 10 小时才能复制 KDS 根密钥并使其可在所有域控制器上使用。

2. 创建 gMSA 账户

要创建 gMSA 账户并允许 `ccg.exe` 检索 gMSA 密码，请从具有域访问权限的 Windows 服务器或客户端上运行以下 PowerShell 命令。将 `ExampleAccount` 替换为您想要的 gMSA 账户名称。

a.
```
PS C:\> Install-WindowsFeature RSAT-AD-PowerShell
```

b.
```
PS C:\> New-ADGroup -Name "ExampleAccount Authorized Hosts" -SamAccountName
 "ExampleAccountHosts" -GroupScope DomainLocal
```

c.
```
PS C:\> New-ADServiceAccount -Name "ExampleAccount" -DnsHostName
 "contoso" -ServicePrincipalNames "host/ExampleAccount", "host/contoso" -
PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
```

d. 使用永不过期的永久密码创建用户。这些凭证存储在 AWS Secrets Manager 中，每个任务都使用这些凭证来加入域。

```
PS C:\> New-ADUser -Name "ExampleAccount" -AccountPassword (ConvertTo-
SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
```

e.
```
PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members
 "ExampleAccount"
```

f. 安装用于在 Active Directory 中创建 CredSpec 对象的 PowerShell 模块并输出 CredSpec JSON。

```
PS C:\> Install-PackageProvider -Name NuGet -Force
```

```
PS C:\> Install-Module CredentialSpec
```

g.
```
PS C:\> New-CredentialSpec -AccountName ExampleAccount
```

3. 将前一个命令的 JSON 输出复制到名为 `gmsa-cred-spec.json` 的文件中。这是 CredSpec 文件。它在步骤 3 步骤 3：修改您的 CredSpec JSON 以包含无域 gMSA 信息 中使用。

步骤 2：将凭证上传到 Secrets Manager

将 Active Directory 凭证复制到安全的凭证存储系统中，以便每个任务都能检索该凭证。这是无域 gMSA 方法。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 `bash`。

- 运行以下 AWS CLI 命令并替换用户名、密码和域名以匹配您的环境。保留密钥的 ARN 以便在下一步 步骤 3：修改您的 CredSpec JSON 以包含无域 gMSA 信息 中使用

  以下命令使用由 `sh` 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleAccount\",\"password\":\"Test123\",
\"domainName\":\"contoso.com\"}"
```

步骤 3：修改您的 CredSpec JSON 以包含无域 gMSA 信息

在将 CredSpec 上传到其中一个存储选项之前，请使用上一步中的 Secrets Manager 中的密钥 ARN 向 CredSpec 添加信息。有关更多信息，请参阅 Microsoft Learn 网站上的非加入域的容器主机应用场景的其他凭证规范配置。

1. 将以下信息添加到 `ActiveDirectoryConfig` 内的 CredSpec 文件。将 ARN 替换为上一步中的 Secrets Manager 中的密钥。

   请注意，`PluginGUID` 值必须与以下示例代码段中的 GUID 相匹配，并且是必填项。

   ```
   "HostAccountConfig": {
         "PortableCcgVersion": "1",
         "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
         "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
   region:111122223333:secret:gmsa-plugin-input\"}"
       }
   ```

   您也可以使用以下格式的 ARN 在 SSM Parameter Store 中使用密钥：`\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`。

2. CredSpec 文件修改后，应类似于以下示例：

   ```
   {
     "CmsPlugins": [
       "ActiveDirectory"
     ],
     "DomainJoinConfig": {
       "Sid": "S-1-5-21-4066351383-705263209-1606769140",
       "MachineAccountName": "ExampleAccount",
       "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
       "DnsTreeName": "contoso",
       "DnsName": "contoso",
       "NetBiosName": "contoso"
     },
     "ActiveDirectoryConfig": {
       "GroupManagedServiceAccounts": [
         {
           "Name": "ExampleAccount",
           "Scope": "contoso"
         },
         {
   ```

```
        "Name": "ExampleAccount",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

步骤 4：将 CredSpec 上传到 Amazon S3

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

1. 将 CredSpec 文件复制到您正在其中运行 AWS CLI 命令的计算机或环境中。
2. 运行以下 AWS CLI 命令以将 CredSpec 上载到 Amazon S3。将 MyBucket 替换为您的 Amazon S3 存储桶的名称。您可以将文件作为对象存储在任何存储桶和位置，但您必须允许访问该存储桶和您附加到任务执行角色的策略中的位置。

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

   ```
   $ aws s3 cp gmsa-cred-spec.json \
   s3://MyBucket/ecs-domainless-gmsa-credspec
   ```

步骤 5：（可选）创建 Amazon ECS 集群

默认情况下，您的账户有一个名为 default 的 Amazon ECS 集群。默认情况下，在 AWS CLI、SDK 和 AWS CloudFormation 中使用此集群。您可以使用其他群集来分组和组织任务和基础设施，并为某些配置分配默认值。

您可以从 AWS Management Console、AWS CLI、SDK 或 AWS CloudFormation 中创建集群。集群中的设置和配置不影响 gMSA。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

```
$ aws ecs create-cluster --cluster-name windows-domainless-gmsa-cluster
```

> **Important**
>
> 如果您选择创建自己的集群，您必须为您打算用于该集群的每个命令指定 `--cluster clusterName`。

步骤 6：为容器实例创建 IAM 角色

容器实例是在 ECS 任务中运行容器的主机，例如 Amazon EC2 实例。每个容器实例都注册到 Amazon ECS 集群。在启动 Amazon EC2 实例并将其注册到集群之前，必须为容器实例创建 IAM 角色以供使用。

要创建容器实例角色，请参阅 Amazon ECS 容器实例 IAM 角色。默认的 `ecsInstanceRole` 有足够的权限完成本教程。

步骤 7：创建自定义任务执行角色

Amazon ECS 可以使用不同的 IAM 角色来获得启动每项任务所需的权限，而不是容器实例角色。此角色称为任务执行角色。建议创建一个仅具有 ECS 运行任务所需的权限（也称为最低权限）的任务执行角色。有关最低权限原则的更多信息，请参阅 AWS Well-Architected Framework 中的 SEC03-BP02 授予最低权限访问。

1. 要创建任务执行角色，请参阅 创建任务执行 角色。默认权限允许容器实例从您的应用程序的 Amazon Elastic Container Registry and `stdout` 和 `stderr` 中提取容器映像，以将其记录到 Amazon CloudWatch Logs 中。

   由于该角色在本教程中需要自定义权限，因此您可以为该角色指定一个不同于 `ecsTaskExecutionRole` 的名称。本教程在后续步骤中使用 `ecsTaskExecutionRole`。
2. 通过创建自定义策略来添加以下权限，可以是仅存在于此角色中的内联策略，也可以是您可以重复使用的策略。将第一条语句中 `Resource` 的 ARN 替换为 Amazon S3 存储桶和位置，将第二个 `Resource` 替换为 Secrets Manager 中密钥的 ARN。

   如果您使用自定义密钥对 Secrets Manager 中的密钥进行加密，则还必须为该密钥允许 `kms:Decrypt`。

   如果您使用 SSM Parameter Store 而不是 Secrets Manager，则必须为该参数允许 `ssm:GetParameter`，而不是 `secretsmanager:GetSecretValue`。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::MyBucket/ecs-domainless-gmsa-credspec/gmsa-cred-
spec.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:aws-region:111122223333:secret:gmsa-
plugin-input"
    }
  ]
}
```

步骤 8：为 Amazon ECS Exec 创建任务角色

本教程使用 Amazon ECS Exec 通过在正在运行的任务中运行命令来验证功能。要使用 ECS Exec，服务或任务必须开启 ECS Exec，并且任务角色（但不是任务执行角色）必须具有 ssmmessages 权限。有关所需的 IAM policy，请参阅 ECS Exec 权限。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

要使用 AWS CLI 创建任务角色，请执行以下步骤。

1. 使用以下内容创建名为 ecs-tasks-trust-policy.json 的文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 创建一个 IAM 角色。您可以替换名称 `ecs-exec-demo-task-role`，但保留该名称以供后续步骤使用。

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

```
$ aws iam create-role --role-name ecs-exec-demo-task-role \
--assume-role-policy-document file://ecs-tasks-trust-policy.json
```

   您可以删除文件 `ecs-tasks-trust-policy.json`。

3. 使用以下内容创建名为 `ecs-exec-demo-task-role-policy.json` 的文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

4. 创建 IAM policy 并将其附加到上一步的角色中。

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

```
$ aws iam put-role-policy \
    --role-name ecs-exec-demo-task-role \
    --policy-name ecs-exec-demo-task-role-policy \
```

```
    --policy-document file://ecs-exec-demo-task-role-policy.json
```

您可以删除文件 ecs-exec-demo-task-role-policy.json。

步骤 9：注册一个使用无域 gMSA 的任务定义

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

1. 使用以下内容创建名为 windows-gmsa-domainless-task-def.json 的文件：

```
{
  "family": "windows-gmsa-domainless-task",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "memory": 1024,
      "essential": true,
      "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::ecs-domainless-gmsa-
credspec/gmsa-cred-spec.json"
      ],
      "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
        }
```

```
      ]
    }
  ],
  "taskRoleArn": "arn:aws:iam::111122223333:role/ecs-exec-demo-task-role",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

2. 通过运行以下命令注册任务定义：

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

```
$ aws ecs register-task-definition \
--cli-input-json file://windows-gmsa-domainless-task-def.json
```

步骤 10：将 Windows 容器实例注册到集群

启动 Amazon EC2 Windows 实例并运行 ECS 容器代理将其注册为集群中的容器实例。ECS 在注册到启动任务的集群的容器实例上运行任务。

1. 要启动在 AWS Management Console 中为 Amazon ECS 配置的 Amazon EC2 Windows 实例，请参阅 启动 Amazon ECS Windows 容器实例。在步骤中停下来获取用户数据。
2. 对于 gMSA，用户数据必须在启动 ECS 容器代理之前设置环境变量 ECS_GMSA_SUPPORTED。

   对于 ECS Exec，代理必须以参数 -EnableTaskIAMRole 开头。

   要通过阻止任务访问 EC2 IMDS Web 服务来检索角色凭证来保护实例 IAM 角色的安全，请添加参数 -AwsvpcBlockIMDS。这仅适用于使用 awsvpc 网络模式的任务。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_GMSA_SUPPORTED", $TRUE, "Machine")
Import-Module ECSTools
Initialize-ECSAgent -Cluster windows-domainless-gmsa-cluster -EnableTaskIAMRole -
AwsvpcBlockIMDS
</powershell>
```

3. 查看 Summary（摘要）面板中您的实例配置的摘要，当您准备好后，选择 Launch instance（启动实例）。

步骤 11：验证容器实例

您可以使用 AWS Management Console 验证集群中是否有容器实例。但是，gMSA 需要以属性表示的其他功能。这些属性在 AWS Management Console 中不可见，因此本教程使用 AWS CLI。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

1. 列出集群中的容器实例。容器实例的 ID 与 EC2 实例的 ID 不同。

   ```
   $ aws ecs list-container-instances
   ```

   输出：

   ```
   {
       "containerInstanceArns": [
           "arn:aws:ecs:aws-region:111122223333:container-
   instance/default/MyContainerInstanceID"
       ]
   }
   ```

   例如，526bd5d0ced448a788768334e79010fd 是有效的容器实例 ID。
2. 使用上一步中的容器实例 ID 获取容器实例的详细信息。将 MyContainerInstanceID 替换为 ID。

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

   ```
   $ aws ecs describe-container-instances \
        ----container-instances MyContainerInstanceID
   ```

   请注意，输出很长。
3. 验证 attributes 列表中是否有一个键名为 name 和值为 ecs.capability.gmsa-domainless 的对象。以下是对象的示例。

   输出：

   ```
   {
       "name": "ecs.capability.gmsa-domainless"
   }
   ```

步骤 12：运行 Windows 任务

运行 Amazon ECS 任务。如果集群中只有 1 个容器实例，则可以使用 `run-task`。如果有许多不同的容器实例，那么使用 `start-task` 和指定运行任务的容器实例 ID 可能比在任务定义中添加放置约束来控制在哪种类型的容器实例上运行该任务要容易得多。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 `bash`。

1. 以下命令使用由 `sh` 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

   ```
   aws ecs run-task --task-definition windows-gmsa-domainless-task \
       --enable-execute-command --cluster windows-domainless-gmsa-cluster
   ```

   记下该命令返回的任何 ID。
2. 运行以下命令验证任务是否已启动。此命令会等待，直到任务开始后才返回 shell 提示符。将 `MyTaskID` 替换为上一步中的任务 ID。

   ```
   $ aws ecs wait tasks-running --task MyTaskID
   ```

步骤 13：验证容器是否有 gMSA 凭证

验证任务中的容器是否有 Kerberos 令牌。gMSA

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 `bash`。

1. 以下命令使用由 `sh` 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

   ```
   $ aws ecs execute-command \
   --task MyTaskID \
   --container windows_sample_app \
   --interactive \
   --command powershell.exe
   ```

   输出将是 PowerShell 提示符。
2. 在容器内的 PowerShell 终端中运行以下命令。

```
PS C:\> klist get ExampleAccount$
```

在输出中，请注意 Principal 是您之前创建的输出。

步骤 14：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。

此步骤使用 AWS CLI。您可以在默认 shell 的 AWS CloudShell 中运行这些命令，即 bash。

1. 停止任务。将 MyTaskID 替换为步骤 12 步骤 12：运行 Windows 任务 中的任务 ID。

   ```
   $ aws ecs stop-task --task MyTaskID
   ```

2. 终止 Amazon EC2 实例。之后，集群中的容器实例将在一小时后自动删除。

   您可以使用 Amazon EC2 控制台查找和终止实例。或者，您可以运行以下命令。要运行该命令，请在步骤 1 步骤 11：验证容器实例 的 aws ecs describe-container-instances 命令输出中找到 EC2 实例 ID。i-10a64379 是 EC2 实例 ID 的示例。

   ```
   $ aws ec2 terminate-instances --instance-ids MyInstanceID
   ```

3. 删除 Amazon S3 中的 CredSpec 文件。将 MyBucket 替换为您的 Amazon S3 存储桶的名称。

   ```
   $ aws s3api delete-object --bucket MyBucket --key ecs-domainless-gmsa-credspec/
   gmsa-cred-spec.json
   ```

4. 在 Secrets Manager 中删除密钥。如果您改用 SSM Parameter Store，请删除该参数。

   以下命令使用由 sh 和兼容的 shell 使用的反斜杠连续字符。此命令与 PowerShell 不兼容。您必须修改该命令才能将其与 PowerShell 配合使用。

   ```
   $ aws secretsmanager delete-secret --secret-id gmsa-plugin-input \
       --force-delete-without-recovery
   ```

5. 取消注册并删除任务定义。通过取消注册任务定义，可以将其标记为非活动状态，这样它就无法用于启动新任务。然后，您可以删除任务定义。

a. 通过指定版本来取消注册任务定义。ECS 会自动生成任务定义的版本，这些版本从 1 开始编号。您可以按照与容器映像上的标签相同的格式来引用版本，例如 `:1`。

```
$ aws ecs deregister-task-definition --task-definition windows-gmsa-domainless-task:1
```

b. 删除任务定义。

```
$ aws ecs delete-task-definitions --task-definition windows-gmsa-domainless-task:1
```

6. （可选）如果您创建了集群，请删除 ECS 集群。

```
$ aws ecs delete-cluster --cluster windows-domainless-gmsa-cluster
```

调试适用于 Windows 容器的 Amazon ECS 无域 gMSA

Amazon ECS 任务状态

ECS 只会尝试启动任务一次。任何有问题的任务都将停止，并设置为状态 `STOPPED`。任务问题有两种常见类型。首先，是无法启动的任务。其次，应用程序已在其中一个容器内停止的任务。在 AWS Management Console 中，查看任务的停止原因字段中，了解任务停止的原因。在 AWS CLI 中，描述任务并查看 `stoppedReason`。有关 AWS Management Console 和 AWS CLI 中的步骤，请参阅 查看 Amazon ECS 已停止任务错误。

Windows 事件

容器中 gMSA 的 Windows 事件记录在 `Microsoft-Windows-Containers-CCG` 日志文件中，可以在 `Logs\Microsoft\Windows\Containers-CCG\Admin` 的“应用程序和服务”部分中的事件查看器中找到。有关更多调试技巧，请参阅 Microsoft Learn 网站上的适用于 Windows 容器的 gMSA 的故障排除。

ECS 代理 gMSA 插件

Windows 容器实例上的 ECS 代理的 gMSA 插件的日志位于以下目录 `C:/ProgramData/Amazon/gmsa-plugin/` 中。查看此日志，以了解无域用户凭证是否是从存储位置（例如 Secrets Manager）下载的，以及凭证格式是否正确读取。

了解了解如何将 gMSA 用于适用于 Amazon ECS 的 EC2 Windows 容器

Amazon ECS 通过组 Managed Service Account (gMSA)的特殊服务账户支持 Windows 容器的 Active Directory 身份验证。

基于 Windows 的网络应用程序（例如 .NET 应用程序）通常使用 Active Directory 来促进用户和服务之间的身份验证和授权管理。开发人员通常将其应用程序设计为与 Active Directory 集成，并为此在加入域的服务器上运行。由于 Windows 容器无法加入域，因此必须将 Windows 容器配置为使用 gMSA 运行。

使用 gMSA 运行的 Windows 容器依赖于其主机 Amazon EC2 实例从 Active Directory 域控制器检索 gMSA 凭据并将其提供给容器实例。有关详细信息，请参阅为 Windows 容器创建 gMSA。

> **Note**
>
> Fargate 上的 Windows 容器不支持此功能。

主题

- 注意事项
- 先决条件
- 在 Amazon ECS 上为 Windows 容器设置 gMSA

注意事项

为 Windows 容器使用 gMSA 时，应考虑以下因素：

- 为容器实例使用经 Amazon ECS 优化的 Windows Server 2016 Full AMI 时，容器主机名必须与凭证规范文件中定义的 gMSA 账户名相同。要为容器指定主机名，请使用 `hostname` 容器定义参数。有关更多信息，请参阅 Network settings (网络设置)。
- 您可以在无域 gMSA 和将每个实例加入单个域之间进行选择。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

  然后，选择数据存储用于 CredSpec，以及无域 gMSA 的 Active Directory 用户凭证（可选）。

Amazon ECS 使用 Active Directory 凭证规范文件（CredSpec）。该文件包含用于将 gMSA 账户上下文传播到容器的 gMSA 元数据。您可以生成 CredSpec 文件，然后将其存储在下表中的一个 CredSpec 存储选项中，该存储选项特定于容器实例的操作系统。要使用无域方法，CredSpec 文件中的可选部分可以在下表中的其中一个 domainless user credentials 存储选项中指定凭证，这些凭证特定于容器实例的操作系统。

按操作系统划分的 gMSA 数据存储选项

存储位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	无域用户凭证	无域用户凭证
Amazon EC2 Systems Manager Parameter Store	CredSpec	CredSpec，无域用户凭证
本地文件	不适用	CredSpec

先决条件

在 Amazon ECS 上为 Windows 容器使用 gMSA 之前，请确保完成以下操作：

- 您可以使用您希望容器访问的资源设置 Active Directory 域。Amazon ECS 支持以下设置：
  - AWS Directory Service Active Directory。AWS Directory Service 是托管在 Amazon EC2 上的 AWS 托管式 Active Directory。有关更多信息，请参阅 AWS Directory Service 管理指南中的 AWS 托管 Microsoft AD 入门。
  - 本地 Active Directory。您必须确保 Amazon ECS Linux 容器实例可以加入域。有关更多信息，请参阅 AWS Direct Connect。
- 您在 Active Directory 中有现有的 gMSA 账户。有关详细信息，请参阅为 Windows 容器创建 gMSA。
- 您选择使用无域 gMSA，或托管 Amazon ECS 任务的 Amazon ECS Windows 容器实例必须是加入到 Active Directory 的域，并且是有权访问 gMSA 账户的 Active Directory 安全组的成员。

  通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

- 您已添加所需的 IAM 权限。所需的权限取决于您为初始凭证和存储凭证规范选择的方法：
  - 如果您使用无域 gMSA 作为初始凭证，则 Amazon EC2 实例角色需要 AWS Secrets Manager 的 IAM 权限。
  - 如果您将凭证规范存储在 SSM Parameter Store 中，则任务执行角色需要获得 Amazon EC2 Systems Manager Parameter Store 的 IAM 权限。
  - 如果您将凭证规范存储在 Amazon S3 中，则任务执行角色需要 Amazon Simple Storage Service 的 IAM 权限。

在 Amazon ECS 上为 Windows 容器设置 gMSA

要在 Amazon ECS 上为 Windows 容器设置 gMSA，您可以按照包括配置先决条件 通过 AWS CLI 将 Amazon ECS Windows 容器与无域 gMSA 结合使用 在内的完整教程进行操作。

以下各节将详细介绍 CredSpec 配置。

主题

- 示例 CredSpec
- 无域 gMSA 设置
- 在任务定义中引用凭证规范文件

示例 CredSpec

Amazon ECS 使用一个凭证规范文件，该文件包含用于将 gMSA 账户上下文传播到 Windows 容器的 gMSA 元数据。您可以生成凭证规范文件并在任务定义的 `credentialSpec` 字段中引用该文件。凭证规范文件不包含任何密钥。

下面是一个示例凭证规范文件：

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "contoso.com",
```

```
        "DnsName": "contoso.com",
        "NetBiosName": "contoso"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "contoso.com"
            }
        ]
    }
}
```

无域 gMSA 设置

建议使用无域 gMSA，而不是将容器实例加入到单个域中。通过使用无域 gMSA，容器实例不会加入该域，实例上的其他应用程序无法使用凭证访问该域，并且加入不同域的任务可以在同一个实例上运行。

1. 在将 CredSpec 上传到其中一个存储选项之前，请在 Secrets Manager 或 SSM Parameter Store 中使用密钥的 ARN 向 CredSpec 添加信息。有关更多信息，请参阅 Microsoft Learn 网站上的非加入域的容器主机应用场景的其他凭证规范配置。

   无域 gMSA 凭证格式

   以下是 Active Directory 的无域 gMSA 凭证的 JSON 格式。将凭证存储在 Secrets Manager 或 SSM Parameter Store 中。

   ```
   {
       "username":"WebApp01",
       "password":"Test123!",
       "domainName":"contoso.com"
   }
   ```

2. 将以下信息添加到 `ActiveDirectoryConfig` 内的 CredSpec 文件。将 ARN 替换为 Secrets Manager 或 SSM Parameter Store 中的密钥。

   请注意，`PluginGUID` 值必须与以下示例代码段中的 GUID 相匹配，并且是必填项。

   ```
       "HostAccountConfig": {
           "PortableCcgVersion": "1",
           "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
   ```

```
        "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
```

您也可以使用以下格式的 ARN 在 SSM Parameter Store 中使用密钥：\"arn:aws:ssm:*aws-region*:111122223333:parameter/*gmsa-plugin-input*\"。

3. CredSpec 文件修改后，应类似于以下示例：

```
{
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "WebApp01",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      },
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

在任务定义中引用凭证规范文件

Amazon ECS 支持使用以下方法在任务定义的 `credentialSpecs` 字段中引用文件路径。对于这些选项中的每一个，您可以提供 `credentialspec:` 或 `domainlesscredentialspec:`，具体取决于您是将容器实例加入单个域还是使用无域 gMSA。

Amazon S3 存储桶

将凭证规范添加到 Amazon S3 存储桶，然后在任务定义的 `credentialSpecs` 字段中引用 Amazon S3 存储桶的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

为了让您的任务可以访问 Amazon S3 存储桶，您还必须将以下权限作为内联策略添加到 Amazon ECS 任务执行 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
```

```
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

SSM Parameter Store 参数

将凭证规范添加到 SSM Parameter Store 参数，然后在任务定义的 `credentialSpecs` 字段中引用 SSM Parameter Store 参数的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [

 "credentialspecdomainless:arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

为了让您的任务可以访问 SSM Parameter Store 参数，您还必须将以下权限作为内联策略添加到 Amazon ECS 任务执行 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ]
```

```
        }
    ]
}
```

本地文件

使用本地文件中的凭证规范详细信息，在任务定义的 credentialSpecs 字段中引用文件路径。引用的文件路径必须相对于 C:\ProgramData\Docker\CredentialSpecs 目录，并使用反斜杠（'\'）作为文件路径分隔符。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspec:file://CredentialSpecDir\CredentialSpecFile.json"
            ],
            ...
        }
    ],
    ...
}
```

使用 EC2 Image Builder 构建经过 Amazon ECS 优化的自定义 AMI

AWS 建议您使用经过 Amazon ECS 优化的 AMI，因为它们预配置了运行容器工作负载的要求和建议。有时您可能需要定制您的 AMI 以添加其他软件。您可以使用 EC2 Image Builder 创建、管理和部署您的服务器映像。您保留在账户中创建的自定义映像的所有权。您可以使用 EC2 Image Builder 管道自动执行映像的更新和系统修补，也可以使用独立命令使用您定义的配置资源创建映像。

您可以为映像创建配方。配方包括父映像和任何其他组件。您还可以创建用于分发自定义 AMI 的管道。

您可以为映像创建配方。Image Builder 映像配方是一个文档，用于定义基础映像和要应用于基础映像以生成输出 AMI 映像所需配置的组件。您还可以创建用于分发自定义 AMI 的管道。有关更多信息，请参阅《EC2 Image Builder 用户指南》中的 EC2 Image Builder 工作原理。

建议您在 EC2 Image Builder 中使用以下经过 Amazon ECS 优化的 AMI 之一作为“父映像”：

- Linux
  - 经 Amazon ECS 优化的 AL2023 x86
  - 经 Amazon ECS 优化的 Amazon Linux 2023（arm64）AMI
  - 经 Amazon ECS 优化的 Amazon Linux 2 内核 5 AMI
  - 经 Amazon ECS 优化的 Amazon Linux 2 x86 AMI
- Windows
  - 经 Amazon ECS 优化的 Windows 2022 Full x86
  - 经 Amazon ECS 优化的 Windows 2022 Core x86
  - 经 Amazon ECS 优化的 Windows 2019 Full x86
  - 经 Amazon ECS 优化的 Windows 2019 Core x86
  - 经 Amazon ECS 优化的 Windows 2016 Full x86

还建议您选择“使用最新可用的 OS 版本”。管道将对父映像使用语义版本控制，这有助于检测自动安排的作业中的依赖项更新。有关更多信息，请参阅《EC2 Image Builder 用户指南》中的语义版本控制。

AWS 使用安全补丁和新的容器代理版本定期更新经 Amazon ECS 优化的 AMI 映像。在映像配方中使用 AMI ID 作为父映像时，您需要定期检查父映像是否有更新。如果有更新，则必须使用更新的 AMI 创建配方的新版本。这样可以确保您的自定义映像包含了最新版本的父映像。有关如何创建工作流以使用新创建的 AMI 自动更新 Amazon ECS 集群中的 EC2 实例的信息，请参阅如何创建 AMI 强化管道并自动更新 ECS 实例集。

您还可以指定通过托管 EC2 Image Builder 管道发布的父映像的 Amazon 资源名称（ARN）。Amazon 会定期通过托管管道发布经 Amazon ECS 优化的 AMI 映像。这些映像可公开访问。您必须具有访问映像的正确权限。当您在 Image Builder 配方中使用映像 ARN 而不是 AMI 时，您的管道在每次运行时都会自动使用父映像的最新版本。通过这种方法，无需为每次更新手动创建新的配方版本。

将映像 ARN 与基础设施即代码（IaC）结合使用

您可以使用 EC2 Image Builder 控制台、基础设施即代码（例如 AWS CloudFormation）或 AWS 开发工具包来配置配方。在配方中指定父映像时，您可以指定 EC2 AMI ID、Image Builder 映像 ARN、AWS Marketplace 产品 ID 或容器映像。AWS 会公开发布经 Amazon ECS 优化的 AMI 的 AMI ID 和 Image Builder 映像 ARN。映像的 ARN 格式如下所示：

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

`ImageVersion` 具有以下格式。将##要、##和##替换为最新值。

```
<major>.<minor>.<patch>
```

您可以将 major、minor 和 patch 替换为特定值，也可以使用映像的无版本 ARN，这样您的管道就可以与最新版本的父映像保持同步。无版本 ARN 使用通配符格式“x.x.x”来表示映像版本。通过这种方法，Image Builder 服务能够自动解析为映像的最新版本。使用无版本 ARN 可确保您的参考始终指向可用的最新映像，从而简化在部署中维护最新映像的过程。当使用控制台创建配方时，EC2 Image Builder 会自动识别您的父映像的 ARN。使用 IaC 创建配方时，您必须识别 ARN 并选择所需的映像版本或使用无版本 ARN，以表示最新的可用映像。建议您创建一个自动脚本，以筛选并仅显示符合您标准的映像。以下 Python 脚本显示如何检索经 Amazon ECS 优化的 AMI 的列表。

该脚本接受两个可选参数：owner 和 platform，其默认值分别为“Amazon”和“Windows”。所有者参数的有效值为：Self、Shared、Amazon 和 ThirdParty。平台参数的有效值为 Windows 和 Linux。例如，如果您在 owner 参数设置为 Amazon 且 platform 设置为 Linux 时运行脚本，则脚本会生成 Amazon 发布的映像列表，包括经 Amazon ECS 优化的映像。

```
import boto3
import argparse

def list_images(owner, platform):
    # Create a Boto3 session
    session = boto3.Session()

    # Create an EC2 Image Builder client
    client = session.client('imagebuilder')

    # Define the initial request parameters
    request_params = {
        'owner': owner,
        'filters': [
            {
                'name': 'platform',
                'values': [platform]
            }
        ]
    }

    # Initialize the results list
    all_images = []

    # Get the initial response with the first page of results
    response = client.list_images(**request_params)
```

```
    # Extract images from the response
    all_images.extend(response['imageVersionList'])

    # While 'nextToken' is present, continue paginating
    while 'nextToken' in response and response['nextToken']:
        # Update the token for the next request
        request_params['nextToken'] = response['nextToken']

        # Get the next page of results
        response = client.list_images(**request_params)

        # Extract images from the response
        all_images.extend(response['imageVersionList'])

    return all_images

def main():
    # Initialize the parser
    parser = argparse.ArgumentParser(description="List AWS images based on owner and
 platform")

    # Add the parameters/arguments
    parser.add_argument("--owner", default="Amazon", help="The owner of the images.
 Default is 'Amazon'.")
    parser.add_argument("--platform", default="Windows", help="The platform type of the
 images. Default is 'Windows'.")

    # Parse the arguments
    args = parser.parse_args()

    # Retrieve all images based on the provided owner and platform
    images = list_images(args.owner, args.platform)

    # Print the details of the images
    for image in images:
        print(f"Name: {image['name']}, Version: {image['version']}, ARN:
 {image['arn']}")

if __name__ == "__main__":
    main()
```

将映像 ARN 与 AWS CloudFormation 结合使用

Image Builder 映像配方是一种蓝图，它指定了实现输出映像预期配置所需的父映像和组件。您使用 AWS::ImageBuilder::ImageRecipe 资源。将 ParentImage 值设置为映像 ARN。使用所需映像的无版本 ARN，确保您的管道始终使用最新版本的映像。例如，arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x。以下 AWS::ImageBuilder::ImageRecipe 资源定义使用 Amazon 托管的映像 ARN：

```
ECSRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: MyRecipe
      Version: '1.0.0'
      Components:
        - ComponentArn: [<The component arns of the image recipe>]
      ParentImage: "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x"
```

有关 AWS::ImageBuilder::ImageRecipe 资源更多信息，请参阅《AWS CloudFormation 用户指南》。

您可以通过设置 AWS::ImageBuilder::ImagePipeline 资源的 Schedule 属性在管道中自动创建新映像。计划包括启动条件和 cron 表达式。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::ImageBuilder::ImagePipeline。

以下 AWS::ImageBuilder::ImagePipeline 示例让管道每天在协调世界时间（UTC）上午 10:00 构建项目。设置以下 Schedule 值：

- 将 PipelineExecutionStartCondition 设置为 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE。只有当依赖资源（例如父映像或在语义版本中使用通配符“x”的组件）更新后，才会启动该版本。这样可以确保版本包含这些资源的最新更新。
- 将 ScheduleExpression 设置为 cron 表达式（0 10 * * ? *）。

```
ECSPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: my-pipeline
```

```
      ImageRecipeArn: <arn of the recipe you created in previous step>
      InfrastructureConfigurationArn: <ARN of the infrastructure configuration
associated with this image pipeline>
      Schedule:
        PipelineExecutionStartCondition:
EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE
        ScheduleExpression: 'cron(0 10 * * ? *)'
```

将映像 ARN 与 Terraform 结合使用

在 Terraform 中指定管道的父映像和计划的方法与 AWS CloudFormation 中的方法一致。您使用 `aws_imagebuilder_image_recipe` 资源。将 `parent_image` 值设置为映像 ARN。使用所需映像的无版本 ARN，确保您的管道始终使用最新版本的映像。有关更多信息，请参阅 Terraform 文档中的 `aws_imagebuilder_image_recipe`。

在 `aws_imagebuilder_image_pipeline resource` 的调度配置块中，将 `schedule_expression` 参数值设置为您选择的 cron 表达式，以指定管道运行的频率，并将 `pipeline_execution_start_condition` 设置为 `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`。有关更多信息，请参阅 Terraform 文档中的 `aws_imagebuilder_image_pipeline`。

使用 Amazon ECS 上的 AWS Deep Learning Containers

AWS Deep Learning Containers 提供了一组 Docker 映像，用于在 Amazon ECS 上培训和服务 TensorFlow 和 Apache MXNet (Incubating) 中的模型。Deep Learning Containers 支持使用 TensorFlow、NVIDIA CUDA（用于 GPU 实例）和 Intel MKL（用于 CPU 实例）库优化环境。Amazon ECR 中提供了 Deep Learning Containers 的容器镜像，在 Amazon ECS 任务定义中参考。您可以将 Deep Learning Containers 与 Amazon Elastic Inference 结合使用，以降低推理成本。

要开始在 Amazon ECS上使用无 Elastic Inference 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 上的 Elastic Inference Deep Learning Containers

> **Note**
>
> 自 2023 年 4 月 15 日起，AWS 不再允许新客户加入 Amazon Elastic Inference（EI），并将帮助现有客户将其工作负载迁移到价格更低廉且性能更出色的选项。2023 年 4 月 15 日之后，新客户将无法在 Amazon SageMaker、Amazon ECS 或 Amazon EC2 中使用 Amazon EI 加速

器启动实例。但是，在过去 30 天内至少使用过一次 Amazon EI 的客户将视为当前客户，可继续使用该服务。

AWS Deep Learning Containers 为 TensorFlow 和 Apache MXNet (Incubating) 中的模型提供了一组 Docker 镜像，这些模型利用了 Amazon Elastic Inference 加速器。Amazon ECS 提供任务定义参数，用于将 Elastic Inference 加速器附加到您的容器。当您在任务定义中指定 Elastic Inference 加速器类型时， Amazon ECS将托管加速器的生命周期和配置。使用此功能时需要 Amazon ECS 服务相关角色。有关 Elastic Inference 加速器的更多信息，请参阅 Amazon Elastic Inference 基础。

Important

您的 Amazon ECS 容器实例需要不低于版本 `1.30.0` Amazon ECS 的容器代理。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。

要开始在 Amazon ECS 上使用具有 Elastic Inference 的 Deep Learning Containers，请参阅 Amazon Elastic Inference 开发人员指南中的 Amazon ECS 上具有 Elastic Inference 的 Deep Learning Containers

Amazon ECS 服务配额

下表提供了 AWS 账户 的 Amazon ECS 默认服务限额（也称为限制）。有关可与 Amazon ECS 结合使用的其他 AWS 服务（例如 Elastic Load Balancing 和 Auto Scaling）的服务限额的更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务限额。有关 Amazon ECS API 中的 API 限制的信息，请参阅请求对 Amazon ECS API 进行限制。

Amazon ECS 服务配额

以下是 Amazon ECS 服务限额。

新 AWS 账户的初始配额可能较低，但会随着时间的推移而增加。Amazon ECS 会持续监控每个区域内的账户使用情况，然后根据您的使用情况自动增加配额。您还可以请求提高显示为可调整的值的配额，请参阅《服务限额用户指南》中的请求提升配额。

名称	默认值	可调整	描述
每个集群的容量提供程序	每个受支持的区域：20 个	否	可以与集群关联的最大容量提供程序数量。
每个服务的经典负载均衡器	每个受支持的区域：1 个	否	每个服务的最大经典负载均衡器数。
每个账户的集群数	每个受支持的区域：10000 个	是	每个每个账户的集群数
每个集群的容器实例数	每个受支持的区域：5,000 个	否	每个集群的容器实例数
每个启动任务的容器实例数	每个受支持的区域：10 个	否	StartTask API 操作中指定的容器实例的最大数量。
每个任务定义的容器数	每个受支持的区域：10 个	否	任务定义中的最大容器定义数。

名称	默认值	可调整	描述
ECS Exec 会话	每个受支持的区域：1000 个	否	每个容器的最大 ECS Exec 会话数。
服务在 AWS Fargate 上启动的任务率	每个受支持的区域：500 个	否	Amazon ECS 服务调度器在 Fargate 上每分钟可以为每个服务预置的最大任务数。
服务在 Amazon EC2 或外部实例上启动的任务率	每个受支持的区域：500 个	否	Amazon ECS 服务调度器在 Amazon EC2 或外部实例上每分钟可以为每个服务预置的最大任务数。
每个任务定义系列的修订数	每个受支持的区域：1000000 个	否	每个任务定义系列的最大修订数。注销或删除任务定义修订不会将其排除在此限制之外。
每个 awsvpcConfiguration 的安全组数	每个受支持的区域：5 个	否	awsvpcConfiguration 中指定的最大安全组数。
每个集群的服务数	每个受支持的区域：5,000 个	否	每个集群的最大服务数
每个命名空间的服务数	每个受支持的区域：100 个	是	在命名空间中运行的服务的最大数量。
每个 awsvpcConfiguration 的子网数	每个受支持的区域：16 个	否	awsvpcConfiguration 中指定的最大子网数。
每个资源的标签	每个受支持的区域：50 个	否	每个资源的最大标签数。这适用于任务定义、集群、任务和服务。

名称	默认值	可调整	描述
每个服务的目标组数	每个受支持的区域：5 个	否	每个服务的最大目标组数（如果使用应用程序负载均衡器或网络负载均衡器）。
任务定义大小	每个受支持的区域：64KB	否	任务定义的最大大小（以 KiB 为单位）。
每个集群处于预置状态的任务	每个受支持的区域：500 个	否	每个集群中等待在预置状态的最大任务数。此配额仅适用于使用 EC2 自动扩缩组容量提供程序启动的任务。
每个运行任务启动的任务数	每个受支持的区域：10 个	否	每个 RunTask API 操作可启动的任务的最大数量。
每个服务的任务数	每个受支持的区域：5,000 个	否	每个服务的最大任务数（预期数量）。

Note

默认值是由 AWS 设置的初始配额，它们与实际应用的配额值和最大可能的服务限额是分开的。有关更多信息，请参阅《服务限额用户指南》中的服务限额中的术语。

Note

配置为使用 Amazon ECS 服务发现的服务的任务限制为每个服务 1000 个任务。这是由于每个服务的实例数量的AWS Cloud Map服务限额有限。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS Cloud Map 服务限额。

> **Note**
>
> 实际上，任务启动率还取决于其他考虑因素，例如要下载和解压的容器镜像、启用的运行状况检查和其他集成，例如向负载均衡器注册任务。您可能会看到任务启动率与此处显示的限额相比的差异。这些变化是由您用于服务的功能引起的。有关更多信息，请参阅 Amazon ECS 服务参数的最佳实践 。

> **Note**
>
> 配置为使用 Amazon ECS Service Connect 的服务的任务限制为每个服务 1000 个任务。这是由于每个服务的实例数量的AWS Cloud Map服务限额有限。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS Cloud Map 服务限额。

AWS Fargate 服务限额

以下是 AWS Fargate 上的 Amazon ECS 服务限额，在服务限额控制台中的 AWS Fargate 服务下列出。

新 AWS 账户的初始配额可能较低，但会随着时间的推移而增加。Fargate 会持续监控每个区域内的账户使用情况，然后根据您的使用情况自动增加配额。您还可以请求提高显示为可调整的值的配额，请参阅《服务限额用户指南》中的请求提升配额。

名称	默认值	可调整	描述
Fargate 按需型 vCPU 资源计数	每个受支持的区域：6 个	是	此账户在当前区域中作为 Fargate 按需型实例并发运行的 Fargate vCPU 数量。
Fargate 竞价型 vCPU 资源计数	每个受支持的区域：6 个	是	此账户在当前区域中作为 Fargate 竞价型实例并发运行的 Fargate vCPU 数量。

> **Note**
>
> 默认值是由 AWS 设置的初始配额，它们与实际应用的配额值和最大可能的服务限额是分开的。有关更多信息，请参阅《服务限额用户指南》中的服务限额中的术语。

> **Note**
>
> Fargate 还强制执行 Amazon ECS 任务和 Amazon EKS 容器组启动率限制。有关更多信息，请参阅 Fargate 限制。

在 AWS Management Console 中管理您的 Amazon ECS 和 AWS Fargate Service Quotas

Amazon ECS 已与 Service Quotas 集成，AWS 服务可让您从中心位置查看和托管您的配额。有关更多信息，请参阅 Service Quotas 用户指南中的什么是 Service Quotas？

使用 Service Quotas，可使用轻松查找 Amazon ECS 服务配额的值。

AWS Management Console

使用 查看 Amazon ECS 和 Fargate 服务配额 AWS Management Console

1. 访问 https://console.aws.amazon.com/servicequotas/，打开服务限额控制台。
2. 在导航窗格中，选择 AWS 服务。
3. 从 AWS 服务列表中，搜索并选择 Amazon Elastic Container Service (Amazon ECS) 或 AWS Fargate。

   在服务限额列表中，您可以查看服务限额名称、应用的值（如果该值可用）、AWS 默认限额以及限额值是否可调整。
4. 要查看有关服务限额的其他信息（如描述），请选择限额名称。
5. （可选）要请求增加配额，请选择要增加的配额，选择 Request quota increase（请求增加配额），输入或选择所需信息，然后选择 Request（请求）。

要使用 AWS Management Console 进一步处理服务配额，请参阅 Service Quotas 用户指南。要请求提高限额，请参阅《服务限额用户指南》中的请求提高限额。

AWS CLI

要使用 AWS CLI 查看 Amazon ECS 和 Fargate 服务配额

运行以下命令查看 Amazon ECS 原定设置配额。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code ecs \
    --output table
```

运行以下命令查看原定设置 Fargate 配额。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

运行以下命令查看您应用的 Fargate 配额。

```
aws service-quotas list-service-quotas \
    --service-code fargate
```

> **Note**
>
> Amazon ECS 不支持应用的限额。

有关使用 AWS CLI 处理服务限额的更多信息，请参阅服务限额 AWS CLI 命令参考。要请求提高配额，请参阅 AWS CLI 命令参考中的 `request-service-quota-increase` 命令。

处理 Amazon ECS 服务配额和 API 节流限制

Amazon ECS 与多个 AWS 服务 集成，包括 Elastic Load Balancing、AWS Cloud Map 和 Amazon EC2。通过这种紧密集成，Amazon ECS 包括多项功能，例如服务负载均衡、Service Connect、任务联网和集群自动扩缩。Amazon ECS 和与之集成的其他 AWS 服务 均保持服务配额和 API 速率限制，

以确保一致的性能和利用率。这些服务配额还可以防止意外预置超出需求的资源，并防止可能增加账单的恶意行为。

通过自行熟悉您的服务配额和 AWS API 速率限制，您可以计划扩缩工作负载，而不必担心性能意外下降。有关更多信息，请参阅请求对 Amazon ECS API 进行节流。

在 Amazon ECS 上扩缩工作负载时，我们建议您考虑以下服务配额。

- AWS Fargate 有限制每个 AWS 区域 中并发运行的任务数的配额。Amazon ECS 上有按需和 Fargate Spot 任务的配额。每个服务配额还包括您在 Fargate 上运行的任何 Amazon EKS Pod。
- 对于在 Amazon EC2 实例上运行的任务，您可以为每个集群注册的最大 Amazon EC2 实例数为 5,000 个。如果您将 Amazon ECS 集群自动扩缩与自动扩缩组容量提供程序一起使用，或者您自己管理集群的 Amazon EC2 实例，则此配额可能会成为部署瓶颈。如果您需要更多容量，则可以创建更多集群或请求提高服务配额。
- 如果您将 Amazon ECS 集群自动扩缩与自动扩缩组容量提供程序一起使用，则在扩缩服务时请考虑 `Tasks in the PROVISIONING state per cluster` 配额。此配额是容量提供程序可以为其增加容量的每个集群处于 `PROVISIONING` 状态时的最大任务数。当您同时启动大量任务时，很容易达到这一配额。例如，如果您同时部署数十项服务，每项服务都有数百个任务。发生这种情况时，容量提供程序需要启动新的容器实例，以便在集群容量不足时下达任务。容量提供程序启动其他 Amazon EC2 实例时，Amazon ECS 服务计划程序很可能会继续并行启动任务。但是，由于集群容量不足，此活动可能会受到限制。Amazon ECS 服务计划程序实施了回退和指数节流策略，用于在新容器实例启动时重试任务放置。因此，您可能会遇到部署速度较慢或横向扩展时间较长的问题。为避免这种情况，您可以通过以下方式之一来规划服务部署。部署大量不需要增加集群容量的任务，或为启动新任务保留备用集群容量。

在扩缩工作负载时，除了考虑 Amazon ECS 的服务配额外，还要考虑与 Amazon ECS 集成的其他 AWS 服务 的服务配额。

Elastic Load Balancing

您可以将 Amazon ECS 服务配置为使用 Elastic Load Balancing 跨任务平均分配流量。有关如何选择负载均衡器的更多信息和推荐的最佳实践，请参阅 使用负载均衡分配 Amazon ECS 服务流量。

Elastic Load Balancing 服务配额

在扩缩工作负载时，请考虑以下 Elastic Load Balancing 服务配额。大多数 Elastic Load Balancing 服务配额都是可调的，您可以在服务配额控制台请求增加配额。

应用程序负载均衡器

使用应用程序负载均衡器时，根据使用案例，您可能需要请求增加以下配额：

- `Targets per Application Load Balancer` 配额，即应用程序负载均衡器背后的目标数量。
- `Targets per Target Group per Region` 配额，即目标组背后的目标数量。

有关更多信息，请参阅《应用程序负载均衡器用户指南》中的应用程序负载均衡器的配额。

Network Load Balancer

您可以向网络负载均衡器注册的目标数量有更严格的限制。使用网络负载均衡器时，您通常需要启用跨区域支持，这对 `Targets per Availability Zone Per Network Load Balancer`（每个网络负载均衡器每个可用区的最大目标数）有额外的扩缩限制。有关更多信息，请参阅《网络负载均衡器用户指南》中的网络负载均衡器的配额。

Elastic Load Balancing API 节流

当您将 Amazon ECS 服务配置为使用负载均衡器时，必须通过目标组运行状况检查才能视为运行正常。为了执行这些运行状况检查，Amazon ECS 会代表您调用 Elastic Load Balancing API 操作。如果您的账户中有大量服务配置了负载均衡器，则可能会减慢服务部署速度，因为可能会有专门针对 `RegisterTarget`、`DeregisterTarget` 和 `DescribeTargetHealth` Elastic Load Balancing API 操作的节流。发生节流时，您的 Amazon ECS 服务事件消息中会出现节流错误。

如果您遇到 AWS Cloud Map API 节流，则可以联系 AWS Support 以获取有关如何提高 AWS Cloud Map API 节流限制的指导。有关监控和排查此类节流错误的更多信息，请参阅处理 Amazon ECS 节流问题。

弹性网络接口

您的任务使用 `awsvpc` 网络模式时，Amazon ECS 会为每个任务预置一个唯一的弹性网络接口（ENI）。当您的 Amazon ECS 服务使用 Elastic Load Balancing 负载均衡器时，这些网络接口也会注册为服务中定义的相应目标组的目标。

弹性网络接口服务配额

当您运行使用 `awsvpc` 网络模式的任务时，每个任务都会附加一个唯一的弹性网络接口。如果必须通过互联网访问这些任务，则请为这些任务的弹性网络接口分配公有 IP 地址。在扩缩 Amazon ECS 工作负载时，请考虑以下两个重要配额：

- `Network interfaces per Region` 配额，即您的账户 AWS 区域 中的最大网络接口数。
- `Elastic IP addresses per Region` 配额，即 AWS 区域 中的最大弹性 IP 地址数。

这两个服务配额均可调，您可以从服务配额控制台请求增加这些配额。有关更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的 Amazon VPC 服务配额。

对于托管在 Amazon EC2 实例上的 Amazon ECS 工作负载，运行使用 awsvpc 网络模式的任务时，请考虑 Maximum network interfaces 服务配额，即每个 Amazon EC2 实例的最大网络实例数。此配额限制了您可以在实例上放置的任务数。您无法调整该配额，并且其在服务配额控制台中不可用。有关更多信息，请参阅《Amazon EC2 用户指南》中的每种实例类型的每个网络接口的 IP 地址数。

尽管您无法更改可以附加到 Amazon EC2 实例的网络接口的数量，但您可以使用弹性网络接口中继功能增加可用的网络接口数。例如，默认情况下，一个 c5.large 实例最多可能有 3 个网络接口。实例的主网络接口计为一个。因此，您可以向该实例再附加两个网络接口。由于每个使用 awsvpc 网络模式的任务都需要一个网络接口，因此，您通常只能在该实例类型上运行两个此类任务。这可能会导致未充分利用集群容量。如果您启用弹性网络接口中继，则可以增加网络接口密度，以在每个实例上放置更多的任务数。开启中继后，一个 c5.large 实例最多可以有 12 个网络接口。实例具有主网络接口，而 Amazon ECS 会创建一个“中继”网络接口并将此接口附加到该实例。因此，使用此配置后，您可以在实例上运行 10 个任务，而不是默认的两个任务。有关更多信息，请参阅 增加 Amazon ECS Linux 容器实例网络接口。

弹性网络接口 API 节流

当您运行使用 awsvpc 网络模式的任务时，Amazon ECS 依赖于以下 Amazon EC2 API。这些 API 都有不同的 API 限制。有关更多信息，请参阅《Amazon EC2 API Reference》中的 Request throttling for the Amazon EC2 API。

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

如果 Amazon EC2 API 调用在弹性网络接口预置工作流程期间受到限制，则 Amazon ECS 服务计划程序会使用指数级回退自动重试。这些自动重试有时可能会导致任务启动延迟，从而导致部署速度变慢。

发生 API 节流时，您将在服务事件消息中看到消息 `Operations are being throttled. Will try again later.`。如果您一直满足 Amazon EC2 API 限制，则可以联系 AWS Support 以获取有关如何提高 API 节流限制的指导。有关监控和排查节流错误的更多信息，请参阅处理节流问题。

AWS Cloud Map

Amazon ECS 服务发现和 Service Connect 使用 AWS Cloud Map API 管理 Amazon ECS 服务的命名空间。如果您的服务有大量任务，则请考虑以下建议。

AWS Cloud Map 服务限额

当 Amazon ECS 服务配置为使用服务发现或 Service Connect 时，`Tasks per service` 配额（即该服务的最大任务数）会受到 AWS Cloud Map `Instances per service` 服务配额（即该服务的最大实例数）的影响。特别是，AWS Cloud Map 服务配额将您可以运行的任务数量减少到最多 10000 个服务任务。您不能更改 AWS Cloud Map 配额。有关更多信息，请参阅AWS Cloud Map服务配额。

AWS Cloud Map API 节流

Amazon ECS 代表您调用 `ListInstances`、`GetInstancesHealthStatus`、`RegisterInstance` 和 `DeregisterInstance` AWS Cloud Map API。其有助于服务发现，并在您启动任务时执行运行状况检查。当同时部署使用服务发现和具有大量任务的多项服务时，这可能会导致超出 AWS Cloud Map API 节流限制。发生这种情况时，您可能会看到以下消息：`Operations are being throttled. Will try again later`（在您的 Amazon ECS 服务事件消息中），并且部署和任务启动速度变慢。AWS Cloud Map 没有记录这些 API 的节流限制。如果您因这些而遇到节流，则可以联系 AWS Support 以获取有关提高 API 节流限制的指导。有关监控和排查此类节流错误的更多建议，请参阅处理 Amazon ECS 节流问题。

Amazon ECS API 参考

除了 AWS Management Console 和 AWS Command Line Interface（AWS CLI）之外，Amazon ECS 还提供了一个 API。您可以使用 API，自动执行管理 Amazon ECS 资源的任务。

- 有关按 Amazon ECS 资源进行的 API 操作列表，请参阅按 Amazon ECS 资源执行的操作。
- 有关 API 操作的字母顺序列表，请参阅操作。
- 有关数据类型的字母顺序列表，请参阅数据类型。
- 有关常用查询参数的列表，请参阅常用参数。
- 有关错误代码的描述，请参阅常见错误。

有关 AWS CLI 的更多信息，请参阅《适用于 Amazon ECS 的 AWS Command Line Interface 参考》https://docs.aws.amazon.com/cli/latest/reference/ecs/index.html。

文档历史记录

下表介绍了 Amazon Elastic Container 开发人员指南的主要更新和新功能。我们还经常更新文档来处理发送给我们的反馈意见。

更改	描述	日期
Fargate 支持上适用于 Linux 容器的 gMSA	Amazon ECS 通过称为组托管服务账户（gMSA）的特殊服务账户支持 Linux 容器的 Active Directory 身份验证。有关更多信息，请参阅将 gMSA 用于 Fargate 上的 Linux 容器。	2024 年 3 月 5 日
为附加到任务的 Amazon EBS 卷添加的 CloudWatch 指标	Amazon ECS 现在针对附加 Amazon EBS 卷的任务发布 Amazon EBS 存储利用率的 CloudWatch 指标。有关更多信息，请参阅 Amazon ECS CloudWatch 指标。	2024 年 2 月 8 日
Service Connect TLS	现在，您可以将 TLS 与 Service Connect 配合使用。	2024 年 1 月 22 日
Service Connect TLS 托管策略	新增 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 策略。	2024 年 1 月 22 日
Service Connect 超时配置更新	现在，可以更新 Service Connect 超时配置，包括两个可选的参数 – `idleTimeout` 和 `perRequestTimeout`。	2024 年 1 月 22 日
Amazon ECS 托管实例耗尽	您可以使用 Amazon ECS 托管实例耗尽功能协助优雅终止 Amazon ECS 实例。	2024 年 1 月 19 日
为 ECS Anywhere 添加了 Ubuntu 22 支持	ECS Anywhere 添加了对 Ubuntu 22 操作系统的支持。有关更多信息，请参阅 支持的操作系统和系统体系结构。	2024 年 1 月 16 日
添加 `AmazonECSInfrastructureRolePolicyForVolumes` IAM 策略	`AmazonECSInfrastructureolePolicyForVolumes` 已添加。该策略授予 Amazon ECS 进行 AWS API 调用所需的权限，以管理与 Amazon ECS 工作负载关联的 Amazon EBS 卷。	2024 年 1 月 11 日

更改	描述	日期
Amazon ECS 任务的 Amazon EBS 数据卷	您可以在任务部署期间为每个任务配置一个 Amazon EBS 数据卷，以便附加到独立的 Amazon ECS 任务或由 ECS 服务托管的任务。在部署时配置卷，允许您可以创建不受特定卷类型或设置约束的可重复使用的任务定义。Amazon EBS 卷为数据密集型容器化工作负载提供高度可用、经济实惠、持久、高性能的块存储。	2024 年 1 月 11 日
Amazon ECS 经典控制台已终止使用	Amazon ECS 控制台已终止使用。	2023 年 12 月 4 日
更新的策略	AmazonECSServiceRolePolicy 托管 IAM 策略已更新，增加了新的 `events` 权限以及附加的 `autoscaling` 和 `autoscaling-plans` 权限。	2023 年 12 月 4 日
运行时监控支持	您可以使用运行时监控来监控您的 Amazon ECS 工作负载，以识别恶意或未经授权的行为。有关更多信息，请参阅运行时监控。	2023 年 11 月 26 日
更新的策略	`AmazonECSServiceRolePolicy` 托管 IAM policy 已更新，允许访问 AWS Cloud Map `DiscoverInstancesRevision` API。	2023 年 10 月 4 日
AWS Fargate 任务停用配置	您可以配置 Fargate 任务停用之前的等待时间。有关更多信息，请参阅 AWS Fargate 任务维护。	2023 年 9 月 5 日
AWS Fargate 中额外的任务定义参数	AWS Fargate 在 Linux 平台版本 `1.4.0` 中添加了对 `pidMode` 和 `systemControls` 的支持。有关更多信息，请参阅任务定义。	2023 年 8 月 9 日
Amazon ECS 控制台任务定义页面重新设计	Amazon ECS 控制台中的任务定义页面已经过重新设计，其中包含额外选项。有关更多信息，请参阅使用控制台创建任务定义。	2023 年 7 月 26 日

更改	描述	日期
Fargate 支持使用 Seekable OCI 索引进行延迟加载	AWS Fargate 正在引入 Seekable OCI（SOCI）索引。借助 SOCI，容器只需花几秒钟的时间进行映像拉取即可启动，从而在后台下载映像时为环境设置和应用程序实例化提供了时间。有关更多信息，请参阅《适用于 AWS Fargate 的 Amazon ECS 用户指南》中的使用 Seekable OCI（SOCI）延迟加载容器映像。	2023 年 7 月 17 日
改进了 Linux 和 Windows 上对 gMSA 的支持	任务定义中有一个适用于 Linux 和 Windows 的 gMSA 的新 `credentialSpecs` 字段。增加了 Windows 上无域 gMSA 的新的完整教程，请参阅教程：使用 AWS CLI 将 Windows 容器与无域 gMSA 结合使用。有关更多信息，请参阅将 gMSA 用于 Linux 容器和将 gMSA 用于 Windows 容器。	2023 年 7 月 14 日
改进了 ECS 代理版本文档	Amazon ECS 代理版本的文档已更新。建议您将 `v20.10.13` 版本或更新版本的 Docker 与最新版本的 Amazon ECS 容器代理结合使用。已发布的版本和对代理的更改可在 GitHub 上找到。有关更多信息，请参阅 Amazon ECS Linux 容器代理版本。	2023 年 6 月 20 日
更新了支持 Fargate ARM64 的区域可用性	已更新支持 Fargate ARM64 的区域可用性。有关更多信息，请参阅 注意事项。	2023 年 6 月 19 日
改进集群自动扩缩文档	Amazon EC2 Auto Scaling 的 Amazon ECS 扩展文档在准确性和可读性方面有了显著改进。有关更多信息，请参阅 Amazon ECS 集群自动扩缩。	2023 年 5 月 4 日
标记授权以进行资源创建。	用户必须具有创建资源的操作的权限，例如 `ecsCreateCluster` 。当您创建资源并为该资源指定标签时，AWS 会执行额外授权以验证是否有创建标签的权限。有关更多信息，请参阅标记授权和在创建时授予标记资源的权限。	2023 年 4 月 18 日
支持 EC2 上的 Linux 容器的 gMSA	您可以使用 gMSA 对 EC2 上的 Linux 容器的 Active Directory 进行身份验证。有关更多信息，请参阅对 Linux 容器使用 gMSA。	2023 年 4 月 14 日

更改	描述	日期
在 AWS Fargate 上支持 Windows 容器的短暂存储	您可以在 AWS Fargate 上使用 Windows 容器的短暂存储。有关更多信息，请参阅 Fargate 任务存储。	2023 年 4 月 14 日
AWS Cost Management 支持任务级 CUR 数据	您可以在成本和使用情况报告中开启任务级成本和资源使用量。此操作为在 AWS Fargate 和 EC2 上运行的任务添加了拆分成本分配数据。有关更多信息，请参阅 任务级成本和使用量报告。	2023 年 4 月 12 日
Amazon Linux 2023 经 Amazon ECS 优化的 AMI	您可以在 Amazon Linux 2023 经 Amazon ECS 优化的 AMI 上部署工作负载。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2023 年 4 月 10 日
AWS Fargate 美国联邦信息处理标准（FIPS）140	您可以按照符合美国联邦信息处理标准（FIPS）140 的方式在 AWS Fargate 上的 Amazon ECS 中部署工作负载。有关更多信息，请参阅 AWS Fargate 美国联邦信息处理标准（FIPS-140）。	2023 年 4 月 10 日
任务定义删除	您可以使用 Amazon ECS 控制台、SDK 和 AWS CLI 删除任务定义。有关更多信息，请参阅使用控制台删除任务定义修订和任务定义。	2023 年 2 月 24 日
Compute Optimizer 中的 AWS Fargate 服务建议	AWS Compute Optimizer 根据 AWS Fargate 上的 Amazon ECS 服务中正在运行的任务的利用率生成任务和容器大小建议。有关更多信息，请参阅查看为 Fargate 上的 Amazon ECS 服务生成的建议。	2023 年 1 月 27 日
Amazon ECS 控制台	新的 Amazon ECS 控制台现在是默认控制台。有关更多信息，请参阅新的 Amazon ECS 控制台。	2023 年 1 月 19 日
更新 `AmazonECS_FullAccess` IAM policy	`AmazonECS_FullAccess` IAM policy 已更新，包括在创建期间向负载均衡器添加标签的权限。有关更多信息，请参阅 AmazonECS_FullAccess。	2023 年 1 月 4 日
使用 CloudWatch 警报检测 Amazon ECS 服务部署故障	您可以配置 Amazon ECS 以在检测到指定的 CloudWatch 警报已进入 ALARM 状态时，将部署设置为失败。有关更多信息，请参阅 the section called “故障检测 ”。	2022 年 12 月 19 日

更改	描述	日期
支持容器端口映射	您可以在绑定到动态映射的主机端口范围的容器上设置端口号范围。有关更多信息，请参阅 the section called “端口映射”。	2022 年 12 月 15 日
Amazon ECS Service Connect 的一般可用性	此功能增加了由 Amazon ECS 服务部署控制的服务发现和服务网格。有关更多信息，请参阅 the section called “ Service Connect”。	2022 年 11 月 27 日
针对任务定义的新 Amazon ECS 控制台体验已更新	新的 Amazon ECS 控制台体验现在包含适用于任务定义的 JSON 编辑器。有关更多信息，请参阅 the section called “使用控制台创建任务定义”。	2022 年 10 月 27 日
针对任务定义的新 Amazon ECS 控制台体验已更新	新的 Amazon ECS 控制台体验现在包含适用于任务定义的 JSON 编辑器。有关更多信息，请参阅 the section called “使用控制台创建任务定义”。	2022 年 10 月 27 日
更新了新的 Amazon ECS 控制台体验	新的 Amazon ECS 控制台体验已更新，增加了服务和任务参数。有关更多信息，请参阅the section called “创建服务” 和the section called “将应用程序作为任务运行”。	2022 年 10 月 7 日
任务元数据端点版本 4 中的新信息	任务元数据端点版本 4 现在包含 VPC ID 和服务名称。有关更多信息，请参阅 the section called “任务元数据终端节点版本 4”。	2022 年 10 月 7 日
新任务定义大小	Fargate 上的 Amazon ECS 现在支持 8 个 vCPU 和 16 个 vCPU 任务大小。有关更多信息，请参阅 the section called “任务大小”。	2022 年 9 月 16 日
ECS CLI 页面已存档	ECS CLI 文档已存档。我们建议使用 AWS Copilot 来满足您的命令行工具需求。有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。	2022 年 9 月 15 日
新的 Fargate 限额	Fargate 正在从基于任务计数的限额过渡到基于 vCPU 的限额。有关更多信息，请参阅 the section called “AWS Fargate 服务限额”。	2022 年 9 月 8 日

更改	描述	日期
支持 Amazon EC2 Auto Scaling 的暖池。	现在，您可以使用 Amazon EC2 Auto Scaling 温池来更快地横向扩展您的应用程序并节省成本。有关更多信息，请参阅 为您的 Amazon ECS Auto Scaling 组配置预初始化的实例。	2022 年 3 月 23 日
在 ECS Anywhere 中支持 Windows 实例。	ECS Anywhere 现在支持 Windows 实例。有关更多信息，请参阅 外部启动类型的 Amazon ECS 集群。	2022 年 3 月 3 日
增加了对外部实例的 ECS Exec 支持	现在，ECS Exec 支持外部实例。有关更多信息，请参阅 使用 ECS Exec 监控 Amazon ECS 容器。	2022 年 1 月 24 日
更新了新的 Amazon ECS 控制台体验	新的 Amazon ECS 控制台体验支持创建和删除集群、更新任务定义以及注销任务定义。有关更多信息，请参阅 创建 Fargate 启动类型的 Amazon ECS 集群、删除 Amazon ECS 集群、使用控制台更新 Amazon ECS 任务定义 和 使用控制台注销 Amazon ECS 任务定义修订。	2021 年 12 月 8 日
更新了新的 Amazon ECS 控制台体验	新的 Amazon ECS 控制台体验支持创建任务定义。有关更多信息，请参阅 使用控制台创建 Amazon ECS 任务定义。	2021 年 11 月 23 日
Amazon ECS 支持将 64 位 ARM 架构用于 Linux。	Amazon ECS 支持将 64 位 ARM CPU 架构用于 Linux 操作系统。有关更多信息，请参阅 the section called “适用于 64 位 ARM 工作负载的任务定义”。	2021 年 11 月 23 日
Amazon ECS 支持 fluentd log-driver-buffer-limit 选项	Amazon ECS 支持 fluentd `log-driver-buffer-limit` 选项。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。	2021 年 11 月 22 日
经 Amazon ECS 优化的 Linux AMI 构建脚本	Amazon ECS 已对用于构建 Amazon ECS 优化版 AMI 的 Linux 变体的构建脚本进行开源。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI 构建脚本。	2021 年 11 月 19 日
容器实例运行状况	Amazon ECS 增加了对容器实例运行状况监控的支持。有关更多信息，请参阅 监控 Amazon ECS 容器实例运行状况。	2021 年 11 月 10 日

更改	描述	日期
支持 Windows Amazon ECS Exec	Amazon ECS Exec 支持 Windows。有关更多信息，请参阅 使用 ECS Exec 监控 Amazon ECS 容器。	2021 年 11 月 1 日
支持 Fargate 上的 Windows 容器。	Amazon ECS 支持 Fargate 上的 Windows 容器。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Windows 平台版本。	2021 年 10 月 28 日
对 Amazon ECS Anywhere 上的外部实例的 GPU 支持	Amazon ECS 支持在任务定义中为在外部实例上运行的任务指定 GPU 要求。有关更多信息，请参阅适用于 GPU 工作负载的 Amazon ECS 任务定义 和将外部实例注册到 Amazon ECS 集群。	2021 年 10 月 8 日
支持 Windows 上的 awsvpc 网络模式	Amazon ECS 支持 Windows 上的 awsvpc 网络模式。有关更多信息，请参阅 为 Amazon ECS 任务分配网络接口。	2021 年 7 月 15 日
Bottlerocket 正式上市	Amazon ECS 支持经 Amazon ECS 优化的 AMI 变体的Bottlerocket 操作系统是作为 AMI 提供的。有关更多信息，请参阅 经 Amazon ECS 优化的 Bottlerocket AMI。	2021 年 6 月 30 日
Amazon ECS 调度任务更新	Amazon EventBridge 在创建触发 Amazon ECS 调度任务的规则时添加了对其他参数的支持。	2021 年 6 月 25 日
Amazon ECS 的 AWS 托管式策略	Amazon ECS 添加了针对服务相关角色的 AWS 托管策略文档。有关更多信息，请参阅 Amazon Elastic Container Service AWS 托管策略。	2021 年 6 月 8 日
AWS CDK入门	添加了有关使用 AWS CDK 和 Amazon ECS 的入门指南。有关更多信息，请参阅 使用 AWS CDK 创建 Amazon ECS 资源。	2021 年 5 月 27 日
Amazon ECS Anywhere	Amazon ECS 增加了对在集群中注册本地服务器或虚拟机 (VM) 的支持。有关更多信息，请参阅 外部启动类型的 Amazon ECS 集群。	2021 年 5 月 25 日

更改	描述	日期
经 Amazon ECS 优化的 Windows Server 20H2 Core AMI	Amazon ECS 增加了对经 Amazon ECS 优化的 Windows Server 20H2 Core AMI 变体的支持。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2021 年 4 月 19 日
Amazon ECS Exec	Amazon ECS 发布了一个新的调试工具，名为 ECS Exec。有关更多信息，请参阅 使用 ECS Exec 监控 Amazon ECS 容器。	2021 年 3 月 15 日
VPC 端点策略支持	Amazon ECS 现在支持 VPC 端点策略 有关更多信息，请参阅 为 Amazon ECS 创建 VPC 端点策略。	2021 年 1 月 11 日
新控制台体验	Amazon ECS 发布了一种新的控制台体验，支持创建或更新服务或运行独立任务。有关更多信息，请参阅使用控制台创建 Amazon ECS 服务 和将应用程序作为 Amazon ECS 任务运行。	2020 年 12 月 28 日
容量提供程序更新	Amazon ECS 增加了对更新现有 自动扩缩组容量提供程序的支持。	2020 年 11 月 23 日
ECS 现在支持适用于 Windows File Server 的 FSx 的 Windows 任务	Amazon ECS 添加了在 Windows 任务定义中指定适用于 Windows File Server 的 Amazon FSx的支持。有关更多信息，请参阅 将 FSx for Windows File Server 卷与 Amazon ECS 结合使用。	2020 年 11 月 11 日
添加 VPC 双堆栈模式支持	Amazon ECS 增加了对在双堆栈模式下使用 VPC 的支持，使用提供对 IPv6 地址支持的 awsvpc 网络模式。有关更多信息，请参阅 在双堆栈模式下使用 VPC。	2020 年 11 月 5 日
任务元数据端点 v4 更新	Amazon ECS 向任务元数据端点 v4 输出添加了其他元数据。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4。	2020 年 11 月 5 日
支持 Local Zones 和 Wavelength Zones	Amazon ECS 增加了对 Local Zones 和 Wavelength Zones 工作负载的支持。有关更多信息，请参阅 Amazon ECS 应用程序位于共享子网、本地区和 Wavelength 区中。	2020 年 9 月 4 日

更改	描述	日期
Bottlerocket AMI 的 Amazon ECS 变体	Bottlerocket 是一个基于 Linux 的开源操作系统，专门由 AWS 用于运行容器。Bottlerocket 操作系统的经 Amazon ECS 优化的 AMI 变体是作为 AMI 提供的，您可以在启动 Amazon ECS 容器实例时使用的。有关更多信息，请参阅 经 Amazon ECS 优化的 Bottlerocket AMI。	2020 年 8 月 31 日
更新了网络速率统计信息的任务元数据端点版本 4	任务元数据端点版本 4 已更新，提供 Amazon ECS 任务的网络速率统计数据，这些任务使用 `awsvpc` 或者 `bridge` 运行至少版本的 Amazon EC2 实例上托管的网络模式 `1.43.0` 的容器代理。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4。	2020 年 8 月 10 日
Fargate 使用情况指标	AWS Fargate 提供 CloudWatch 使用情况指标，通过该指标可以了解您对 Fargate 按需和 Fargate Spot 资源的账户使用情况。有关更多信息，请参阅 使用情况指标。	2020 年 8 月 3 日
AWS Copilot 0.1.0 版本	新 AWS Copilot CLI启动，提供高级命令，以简化Amazon ECS上本地开发环境中容器化应用程序的建模、创建、发布和管理。有关更多信息，请参阅 使用 AWS Copilot 命令行界面创建 Amazon ECS 资源。	2020 年 7 月 9 日
AWS Fargate 平台版本弃用计划	已添加 Fargate 平台版本弃用计划。有关更多信息，请参阅 AWS Fargate Linux 平台版本弃用。	2020 年 7 月 8 日
AWS Fargate 区域扩展	在 AWS Fargate 上的 Amazon ECS 已经扩展到欧洲（米兰）区域。	2020 年 6 月 25 日
经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI 已发布	Amazon ECS 针对推理工作负载发布了经 Amazon ECS 优化的 Amazon Linux 2（Neuron）AMI。 有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2020 年 6 月 24 日
增加了对容量提供程序删除的支持	Amazon ECS 增加了对 自动扩缩组容量提供程序删除的支持。	2020 年 6 月 11 日

更改	描述	日期
AWS Fargate 平台版本 1.4.0 更新	从 2020 年 5 月 28 日开始，使用平台版本 1.4.0 启动的任何新的 Fargate 任务都将使用 AES-256 加密算法通过 AWS Fargate 托管加密密钥来加密其 20 GB 的短暂存储空间。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate 临时存储。	2020 年 5 月 28 日
环境变量文件支持	添加了对在任务定义中指定环境变量文件的支持，从而使您能够将环境变量批量添加到容器中。有关更多信息，请参阅 将单个环境变量传递给 Amazon ECS 容器。	2020 年 5 月 18 日
AWS Fargate 区域扩展	与 Amazon ECS 结合使用的 AWS Fargate 已经扩展到非洲（开普敦）区域。	2020 年 5 月 11 日
更新了服务配额	更了以下服务配额： • 每个账户的集群数从 2,000 提高到 10,000。 有关更多信息，请参阅 Amazon ECS 服务配额。	2020 年 4 月 17 日

更改	描述	日期
AWS Fargate 平台版本 1.4.0	AWS Fargate 平台版本 1.4.0 已发布，此版本包含以下功能： • 增加了对将 Amazon EFS 文件系统卷用于持久性任务存储的支持。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。 • 短暂任务存储已增至 20 GB。有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate 临时存储。 • 针对任务和来自任务的网络流量行为已更新。从平台版本 1.4 开始，所有 Fargate 任务都会接收单个弹性网络接口（称为任务 ENI），所有网络流量都将流经 VPC 内的这个 ENI，并将通过 VPC 流日志对您可见。有关更多信息，请参阅 Amazon Elastic Container Service 用户指南 AWS Fargate 中的 Fargate 任务联网。 • 任务 ENI 增加对巨型帧的支持。网络接口配置了最大传输单元 (MTU)，这是单个帧内将放入的最大负载的大小。MTU 越大，单个帧内可以放入的应用程序负载就越多，这可以减少每帧开销并提高效率。当您的任务和目标之间的网络路径支持巨型帧时，支持巨型帧将减少开销，如保留在您的 VPC 中的所有流量。 • CloudWatch Container Insights 将包括 Fargate 任务的网络性能指标。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。 • 增加了对任务元数据端点 v4 的支持，该端点为您的 Fargate 任务提供附加信息，包括任务的网络统计信息以及任务所运行的可用区。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4。 •	2020 年 4 月 8 日

更改	描述	日期
	增加了对容器定义中的 SYS_PTRACE Linux 参数的支持。有关更多信息，请参阅 Linux 参数。 • Amazon ECS 容器代理替代了对所有 Fargate 任务使用 Amazon ECS 容器代理。此更改应该不会影响任务的运行方式。 • 容器运行时现在使用 Containerd 而不是 Docker。此更改应该不会影响任务的运行方式。您会注意到，一些源自容器运行时的错误消息将从所提到的 Docker 变为更一般的错误。 有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。	
面向任务卷的 Amazon EFS 文件系统支持	Amazon EFS 文件系统可用作 Amazon ECS 和 Fargate 任务的数据卷。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。	2020 年 4 月 8 日
Amazon ECS 任务元数据端点版本 4	从 Amazon ECS 容器代理版本 1.39.0 和 Fargate 平台版本 1.4.0 开始，名为 ECS_CONTAINER_METADATA_URI_V4 的环境变量被注入到任务中的每个容器中。在您查询任务元数据版本 4 终端节点时，将为任务提供各种任务元数据和 Docker 统计数据。有关更多信息，请参阅 Amazon ECS 任务元数据端点版本 4。	2020 年 4 月 8 日
支持将特定版本的 Secrets Manager 密钥注入为环境变量	添加了对使用特定版本的 Secrets Manager 密钥指定敏感数据的支持。有关更多信息，请参阅 将敏感数据传递给 Amazon ECS 容器。	2020 年 2 月 24 日
为蓝色/绿色部署增加了额外的 CodeDeploy 部署配置选项	CodeDeploy 服务为 Amazon ECS 部署类型添加了新的 Canary 和线性部署配置。还可以定义自定义的部署配置。有关更多信息，请参阅 在部署前验证 Amazon ECS 服务的状态 。	2020 年 2 月 6 日

更改	描述	日期
添加了 efsVolume Configuration 任务定义参数	`efsVolumeConfiguration` 任务定义参数处于公开预览状态，这样可以更轻松地将 Amazon EFS 文件系统用于您的 Amazon ECS 任务。有关更多信息，请参阅 将 Amazon EFS 卷与 Amazon ECS 结合使用。	2020 年 1 月 17 日
Amazon ECS 容器代理日志记录行为已更新	Amazon ECS 容器代理日志记录位置和轮换行为已更新。有关更多信息，请参阅 Amazon ECS 容器代理日志配置参数。	2020 年 1 月 13 日
Fargate Spot	Amazon ECS 增加了对使用 Fargate Spot 运行任务的支持。有关更多信息，请参阅 Fargate 启动类型的 Amazon ECS 集群 。	2019 年 12 月 3 日
集群 Auto Scaling	Amazon ECS 集群 Auto Scaling 使您能够更好地控制集群内的任务调整方式。有关更多信息，请参阅 通过集群自动扩缩功能自动管理 Amazon ECS 容量。	2019 年 12 月 3 日
集群容量提供程序	Amazon ECS 集群容量提供程序确定您的任务要使用的基础设施。有关更多信息，请参阅 Amazon ECS 集群。	2019 年 12 月 3 日
在 AWS Outposts 上创建集群	Amazon ECS 现在支持在 AWS Outposts 上创建集群 有关更多信息，请参阅 the section called “AWS Outposts 上的 Amazon Elastic Container Service”。	2019 年 12 月 3 日
服务操作事件	现在，当发生某些服务操作时，Amazon ECS 会向 Amazon EventBridge 发送事件。有关更多信息，请参阅 Amazon ECS 服务操作事件。	2019 年 11 月 25 日
Amazon ECS 经 GPU 优化的 AMI 支持 G4 实例	Amazon ECS 增加了在使用 Amazon ECS 经 GPU 优化的 AMI 时对 g4 实例类型系列的支持。有关更多信息，请参阅 适用于 GPU 工作负载的 Amazon ECS 任务定义。	2019 年 10 月 8 日

更改	描述	日期
FireLens for Amazon ECS	FireLens for Amazon ECS 已全面推出。FireLens for Amazon ECS 使您可以使用任务定义参数将日志路由到 AWS 服务或合作伙伴目标，以进行日志存储和分析。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。	2019 年 9 月 30 日
AWS Fargate 区域扩展	和Amazon ECS接合使用的 AWS Fargate 已扩展到欧盟（巴黎）、欧盟（斯德哥尔摩）和中东（巴林）区域。	2019 年 9 月 30 日
Amazon ECS 上的 Elastic Inference Deep Learning Containers	Amazon ECS 支持将 Amazon Elastic Inference 加速器连接到您的容器，以使运行深度学习推理工作负载更高效。有关更多信息，请参阅 Amazon ECS 上的 Elastic Inference Deep Learning Containers。	2019 年 9 月 3 日
FireLens for Amazon ECS	FireLens for Amazon ECS 处于公开预览状态。FireLens for Amazon ECS 使您可以使用任务定义参数将日志路由到 AWS 服务或合作伙伴目标，以进行日志存储和分析。有关更多信息，请参阅 将 Amazon ECS 日志发送到 AWS 服务或 AWS Partner。	2019 年 8 月 30 日
CloudWatch Container Insights	CloudWatch Container Insights 现已全面推出。它让您能够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。有关更多信息，请参阅 使用 Container Insights 监控 Amazon ECS 容器。	2019 年 8 月 30 日
容器级别交换配置	Amazon ECS 新增支持，可在容器级别控制 Linux 容器实例上的交换内存空间使用量。使用每容器交换配置，任务定义中的每个容器都可以启用或禁用交换，对于启用交换的容器，可以对所用的最大交换空间量进行限制。有关更多信息，请参阅 管理 Amazon ECS 上的容器交换内存空间。	2019 年 8 月 16 日
AWS Fargate 区域扩展	与 Amazon ECS 接合使用的 AWS Fargate 已扩展到亚太地区（香港）区域。	2019 年 8 月 6 日

更改	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 增加的 Amazon ECS 容器网络接口支持的实例。	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 将多个目标组注册到 Amazon ECS 服务。	2019 年 7 月 30 日
使用 Secrets Manager 密钥指定敏感数据	添加了使用 Secrets Manager 密钥指定敏感数据的教程 有关更多信息，请参阅 在 Amazon ECS 中指定使用 Secrets Manager 密钥的敏感数据。	2019 年 7 月 20 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 使用 Container Insights 监控 Amazon ECS 容器。	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 如何将 Amazon Elastic Container Service 与 IAM 结合使用。	2019 年 6 月 27 日
针对 AWS-2019-005 进行修补的新的经 Amazon ECS 优化的 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 增加 Amazon ECS Linux 容器实例网络接口。	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。	2019 年 5 月 1 日

更改	描述	日期
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 将敏感数据传递给 Amazon ECS 容器。	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 将敏感数据传递给 Amazon ECS 容器。	2019 年 4 月 2 日
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、容器依赖项 和 容器超时。	2019 年 3 月 27 日
Amazon ECS 引入了外部部署类型	外部 部署类型允许您使用任何第三方部署控制器，以完全控制 Amazon ECS 服务的部署过程。有关更多信息，请参阅 使用第三方控制器部署 Amazon ECS 服务。	2019 年 3 月 27 日
Amazon ECS 上的 AWS Deep Learning Containers	AWS Deep Learning Containers 是 Amazon Elastic Container Service (Amazon ECS) 上 TensorFlow 中训练和服务模型的一组 Docker 映像。Deep Learning Containers 提供具有 TensorFlow、Nvidia CUDA（适用于 GPU 实例）和 Intel MKL（适用于 CPU 实例）库的优化环境，其在 Amazon ECR 中可用。有关更多信息，请参阅 使用 Amazon ECS 上的 AWS Deep Learning Containers。	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。	2019 年 3 月 7 日

更改	描述	日期
Amazon ECS 引入 `PutAccountSettingDefault` API	Amazon ECS 推出了 `PutAccountSettingDefault` API，该 API 允许用户为账户上的所有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账户拥有者。 有关更多信息，请参阅 Amazon Resource Names (ARN) 和 ID。	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 适用于 GPU 工作负载的 Amazon ECS 任务定义。	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 将敏感数据传递给 Amazon ECS 容器。	2019 年 1 月 21 日
接口 VPC 端点 (AWS PrivateLink)	添加了对配置由 AWS PrivateLink 提供支持的接口 VPC 端点的支持。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 接口 VPC 端点 (AWS PrivateLink)。	2018 年 12 月 26 日

更改	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 将敏感数据传递给 Amazon ECS 容器。 • 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 Amazon Elastic Container Service 开发人员指南 AWS Fargate 中的任务维护。 有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。	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 服务配额。	2018 年 12 月 14 日
AWS Fargate 区域扩展	与 Amazon ECS 结合使用的 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 Amazon ECS 在 AWS Fargate 上的支持区域。	2018 年 12 月 7 日

更改	描述	日期
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 在部署前验证 Amazon ECS 服务的状态 。	2018 年 11 月 27 日
经 Amazon ECS 优化的 Amazon Linux 2 (arm64) AMI 已发布	Amazon ECS 发布了经 Amazon ECS 优化的 Amazon Linux 2 AMI，适用于 arm64 架构。 有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2018 年 11 月 26 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 • PID 模式	2018 年 11 月 16 日
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 将敏感数据传递给 Amazon ECS 容器。	2018 年 11 月 15 日
为资源加标签	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 为 Amazon ECS 资源添加标签。	2018 年 11 月 15 日
AWS Fargate 区域扩展	与 Amazon ECS 结合使用的 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 适用于 Amazon ECS 的 AWS Fargate。	2018 年 11 月 7 日

更改	描述	日期
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 服务配额。	2018 年 10 月 31 日
AWS Fargate 区域扩展	与 Amazon ECS AWS Fargate 结合使用的 已扩展到欧洲（伦敦）区域。 有关更多信息，请参阅 适用于 Amazon ECS 的 AWS Fargate。	2018 年 10 月 26 日
经 Amazon ECS 优化的 Amazon Linux 2 AMI 已发布	Amazon ECS 提供了已针对服务进行优化的 Linux AMI，有两种变体。最新的推荐版本基于 x;。Amazon ECS 还提供基于 的 AMI，但建议您将工作负载迁移到 Amazon Linux 2 版本，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 经 Amazon ECS 优化的 Linux AMI。	2018 年 10 月 18 日
Amazon ECS 任务元数据端点版本 3	从 Amazon ECS 容器代理版本 1.21.0 开始，代理将调用 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 端点时，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 端点上使用 awsvpc 网络模式的任务。有关更多信息，请参阅 使用 Amazon ECS 元数据监控工作负载。	2018 年 10 月 18 日

更改	描述	日期
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 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。	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 • 交互式 • 伪终端	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 在 Amazon ECS 中使用非 AWS 容器映像。	2018 年 10 月 10 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 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。	2018 年 8 月 30 日
调度任务与 Fargate 任务支持	Amazon ECS 为 Fargate 启动类型引入了对计调度任务的支持。	2018 年 8 月 28 日

更改	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 在 Amazon ECS 中使用非 AWS 容器映像。	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 Amazon ECS 任务的存储选项。	2018 年 8 月 9 日
AWS Fargate 区域扩展	与Amazon ECS 结合使用的 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 适用于 Amazon ECS 的 AWS Fargate。	2018 年 7 月 19 日

更改	描述	日期
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：副本计划策略在集群中放置和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。有关更多信息，请参阅 副本策略。 • 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。有关更多信息，请参阅 进程守护程序策略。 Note Fargate 任务不支持 DAEMON 计划策略。 .	2018 年 6 月 12 日
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。 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 Service Discovery 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 service-discovery 的支持。有关更多信息，请参阅 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。	2018 年 5 月 22 日

更改	描述	日期
添加了对其他经 Amazon ECS 优化的 AMI 元数据参数的支持	添加了子参数，允许您以编程方式检索经 Amazon ECS 优化的 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 检索经 Amazon ECS 优化的 Linux AMI 元数据。	2018 年 5 月 9 日
AWS Fargate 区域扩展	与 Amazon ECS 结合使用的 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 适用于 Amazon ECS 的 AWS Fargate。	2018 年 4 月 26 日
经 Amazon ECS 优化的 AMI 元数据检索	添加了使用 Systems Manager Parameter Store API 以编程方式检索经 Amazon ECS 优化的 AMI 元数据的功能。有关更多信息，请参阅 检索经 Amazon ECS 优化的 Linux AMI 元数据。	2018 年 4 月 10 日
AWS Fargate 平台版本	新 AWS Fargate 平台版本已发布，其中包含： • 增加了对 使用 Amazon ECS 元数据监控工作负载 的支持。 • 增加了对 运行状况检查 的支持。 • 增加了对 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务 的支持 有关更多信息，请参阅 适用于 Amazon ECS 的 Fargate Linux 平台版本。	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 使用服务发现连接具有 DNS 名称的 Amazon ECS 服务。	2018 年 3 月 22 日

更改	描述	日期
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。	2018 年 3 月 20 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。	2018 年 3 月 8 日
AWS Fargate	增加了有关与 AWS Fargate 结合使用的 Amazon ECS 的概述。有关更多信息，请参阅 适用于 Amazon ECS 的 AWS Fargate。	2018 年 2 月 22 日
Amazon ECS 任务元数据端点	从 1.17.0 版 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 端点上使用 awsvpc 网络模式的任务。有关更多信息，请参阅 使用 Amazon ECS 元数据监控工作负载。	2018 年 2 月 8 日
使用目标跟踪策略的 Amazon ECS Service 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 使用目标指标值扩展您的 Amazon ECS 服务。 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
Docker 17.09 支持	添加对 Docker 17.09 的支持。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2018 年 1 月 18 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。	2018 年 1 月 11 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
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

更改	描述	日期
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 policy 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 以拉取容器镜像 • 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。	2017 年 12 月 7 日
Windows 容器支持 GA	增加了对 Windows Server 2016 容器的支持。有关更多信息，请参阅 经 Amazon ECS 优化的 AMI 变体。	2017 年 12 月 5 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。	2017 年 11 月 29 日
Amazon ECS 名称更改	重命名 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 EC2 启动类型的 Amazon ECS 任务联网选项。	2017 年 11 月 14 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。	2017 年 11 月 2 日

更改	描述	日期
Docker 17.06 支持	添加对 Docker 17.06 的支持。有关更多信息，请参阅经 Amazon ECS 优化的 Linux AMI。	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
Network Load Balancer 支持	Amazon ECS 在 Amazon ECS 控制台中增加了对 Network Load Balancers 的支持。	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 将应用程序作为 Amazon ECS 任务运行。	2017 年 6 月 27 日
Amazon ECS 调度任务	增加了对使用 cron 计划任务的支持。	2017 年 6 月 7 日
Amazon ECS 控制台中的 Spot Instances	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 Linux 容器实例。	2017 年 6 月 6 日
新的经 Amazon ECS 优化的 AMI 版本的 Amazon SNS 通知	增加了订阅有关新的经 Amazon ECS 优化的 AMI 版本的 SNS 通知的功能。	2017 年 3 月 23 日
微服务和批处理作业	增加了 Amazon ECS 的两个常见用例 (微服务和批处理作业) 的文档。有关更多信息，请参阅 Amazon ECS 相关信息。	2017 年 2 月

更改	描述	日期
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 耗尽 Amazon ECS 容器实例。	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 经 Amazon ECS 优化的 Linux AMI。	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 使用策略来定义 Amazon ECS 任务放置。	2016 年 12 月 29 日
Windows 容器支持测试版	增加了对 Windows Server 2016 容器的支持 (测试版)。有关更多信息，请参阅 经 Amazon ECS 优化的 AMI 变体。	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 在 Amazon ECS 上计划您的容器。	2016 年 12 月 1 日
针对 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 使用 EventBridge 自动响应 Amazon ECS 错误。	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 将 Amazon ECS 日志发送到 CloudWatch 。	2016 年 9 月 12 日
具有针对动态端口的 Elastic Load Balancing 支持的 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 使用负载均衡分配 Amazon ECS 服务流量。	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 Amazon ECS 任务 IAM 角色。	2016 年 7 月 13 日

更改	描述	日期
Docker 1.11 支持	添加对 Docker 1.11 的支持。有关更多信息，请参阅 经 Amazon ECS 优化的 Linux AMI。	2016 年 5 月 31 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 自动扩展 Amazon ECS 服务。	2016 年 5 月 18 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
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 经 Amazon ECS 优化的 Linux AMI。	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 经 Amazon ECS 优化的 Linux AMI。	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
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
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自动扩缩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

更改	描述	日期
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 Management Console、AWS CLI 和开发工具包更新 ECS 代理。	2015 年 6 月 11 日
Amazon ECS 服务调度程序和 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