Application Auto Scaling

用户指南

Application Auto Scaling: 用户指南

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅中国的 Amazon Web Services 服务入门。

Table of Contents

什么是 Application Auto Scaling？

Application Auto Scaling 是一项 Web 服务，可为开发人员和系统管理员提供一个解决方案，用于弹性伸缩 Amazon EC2 以外的各 Amazon 服务的可扩展资源。Application Auto Scaling 可让您为以下资源配置弹性伸缩：

- Aurora 副本
- DynamoDB 表和全局二级索引
- Amazon Elastic Container Service (ECS) 服务
- Amazon EMR 集群
- Amazon Keyspaces (for Apache Cassandra) 表
- Lambda 函数预置并发
- Amazon Managed Streaming for Apache Kafka (MSK) 代理存储
- Amazon Neptune 集群
- SageMaker 终端节点变体
- Spot 队列请求
- 由您自己的应用程序或服务提供的自定义资源。有关更多信息，请参阅 GitHub 存储库。

要查看上面列出的任意 Amazon 服务的区域可用性，请参阅区域表。

有关使用 Auto Scaling 组扩缩 Amazon EC2 实例队列的信息，请参阅 Amazon EC2 Auto Scaling 用户指南。

您还可以使用 Amazon Auto Scaling 创建扩缩计划，以跨多项服务扩展资源。有关更多信息，请参阅《Amazon Auto Scaling 用户指南》。

Application Auto Scaling 的功能

Application Auto Scaling 可以让您根据您定义的条件弹性伸缩可扩展资源。

- 目标跟踪扩缩 - 根据特定 CloudWatch 指标的目标值扩缩资源。
- 步进扩缩 - 根据一组扩缩调整来扩缩资源，这些调整因警报违例大小而异。
- 计划的扩缩— 仅扩展一次或按经常性计划扩缩资源。

使用 Application Auto Scaling

您可以使用以下界面配置扩缩，具体取决于要扩缩的资源：

- Amazon Web Services Management Console – 提供可用于配置扩缩的 Web 界面。如果您已注册 Amazon 账户，请通过登录 Amazon Web Services Management Console 访问 Application Auto Scaling。然后，打开服务控制台以查看简介中列出的资源之一。请确保在与要使用的资源相同的 Amazon Web Services 区域 中打开控制台。

  Note

  并非所有资源都可以访问控制台。有关更多信息，请参阅可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7)。

- Amazon Command Line Interface（Amazon CLI） – 提供了适用于各种Amazon Web Services的命令，支持 Windows、macOS 和 Linux 等操作系统。要开始使用，请参阅 Amazon Command Line Interface 用户指南。有关更多信息，请参阅 Amazon CLI 命令参考中的 application-autoscaling。
- Amazon Tools for Windows PowerShell – 为在 PowerShell 环境中编写脚本的用户提供大量 Amazon 产品的相关命令。要开始使用，请参阅 Amazon Tools for Windows PowerShell 用户指南。有关更多信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考。
- Amazon SDK – 提供了特定于语言的 API 操作，并简化了许多连接详细信息，例如计算签名、处理请求重试和处理错误。有关更多信息，请参阅 Amazon 开发工具包。
- 查询 API – 提供了您使用 HTTPS 请求调用的低级别 API 操作。使用查询 API 是访问 Amazon Web Services 的最直接方式。但它需要您的应用程序处理低级别的详细信息，例如生成哈希值以签署请求以及处理错误。有关更多信息，请参阅 Application Auto Scaling API 参考。
- Amazon CloudFormation – 支持使用 CloudFormation 模板配置扩缩。有关更多信息，请参阅使用 Amazon CloudFormation 创建 Application Auto Scaling 资源 (p. 99)。

要通过编程方式连接到某个Amazon Web Service，您需要使用终端节点。有关 Application Auto Scaling 调用的端点的信息，请参阅 中的 Application Auto Scaling 端点和配额、中国内的 Amazon Web Services 入门中的中国内的 Amazon Web Services 的端点和 ARN、中

设置

在使用 Application Auto Scaling 配置弹性伸缩之前，请创建 Amazon Web Services 账户、配置访问权限和设置 Amazon Command Line Interface (Amazon CLI)。

主题

注册 Amazon Web Services 账户

当您使用 Amazon Web Services 注册账户时，您的账户会自动注册所有 Amazon，包括 Application Auto Scaling。您只需为使用的服务付费。

如果您还没有 Amazon Web Services 账户，则需要创建一个。如果您已有 Amazon Web Services 账户，则可以跳过以下过程中创建 Amazon Web Services 账户 的步骤。

注册 Amazon Web Services 账户

1. 打开 https://aws.amazon.com/ 并选择 Sign Up（注册）。
2. 按照屏幕上的说明进行操作。在注册过程中，您将接到一通电话，要求您使用电话键盘输入一个验证码。注册过程完成后，Amazon 会向您发送一封确认电子邮件。

在 Amazon Web Services 区域 中使用 Application Auto Scaling

Application Auto Scaling 在多个 Amazon Web Services 区域 中可用。利用全球 Amazon Web Services 账户，您可以在大多数区域中使用资源。当利用中国区域的资源使用 Application Auto Scaling 时，请记住您必须拥有单独的 Amazon Web Services（中国）账户。此外，Application Auto Scaling 的实现方式上存在一些差异。有关在中国区域中使用 Application Auto Scaling 的更多信息，请参阅中国的 Application Auto Scaling。

设置 Amazon CLI

Amazon Command Line Interface (Amazon CLI) 是一款用于管理 Amazon 服务的统一开发工具，包括 Application Auto Scaling。按照以下步骤下载和配置 Amazon CLI。

设置 Amazon CLI

1. 下载并配置 Amazon CLI。有关说明，请参阅《Amazon Command Line Interface 用户指南》中的以下主题：

   - 安装或更新最新版本的 Amazon CLI
   - 快速设置

   Note

   尽管您可以使用与创建 Amazon 账户时提供的电子邮件地址和密码（我们称之为 Amazon 账户根用户）关联的凭证来配置 Amazon CLI，但这并非 Amazon 安全最佳实践。我们建议您

使用 Amazon 账户中的 IAM 管理员用户的凭证配置 Amazon CLI。IAM 管理员用户具有与 Amazon 账户根用户类似的 Amazon 访问权限，并避免一些关联的安全风险。如果无法以 IAM 管理员用户身份配置 Amazon CLI，请与 Amazon 账户管理员联系。有关更多信息，请参阅 IAM 用户指南中的创建您的第一个 IAM 管理员用户和用户组。

2. 为了确认 Amazon CLI 配置文件的配置正确无误，请在命令窗口中运行以下命令：

```
aws configure
```

如果您的配置文件已正确配置，您应该看到类似于以下内容的输出。

```
AWS Access Key ID [****************52FQ]:
AWS Secret Access Key [****************xgyZ]:
Default region name [us-east-1]:
Default output format [json]:
```

3. 运行以下命令确认是否安装了 Amazon CLI 的 Application Auto Scaling 命令。

```
aws application-autoscaling help
```

开始使用 Application Auto Scaling

本主题介绍关键概念，帮助您了解 Application Auto Scaling 并开始使用它。

可扩展目标

您创建的实体，用于指定要扩展的资源。每个可扩展目标都由服务命名空间、资源 ID 和可扩展维度（表示基础服务的一些容量维度）唯一标识。例如，Amazon ECS 服务支持弹性伸缩其任务计数，DynamoDB 表支持弹性伸缩该表及其全局二级索引的读写容量，Aurora 集群支持扩缩其副本计数。

Tip

每个可扩展目标还具有最小容量和最大容量。扩缩策略永远不会高于或低于最小最大范围。您可以直接对超出此范围的基础资源进行带外更改，Application Auto Scaling 不知道这些资源。但是，无论何时调用扩缩策略或调用 `RegisterScalableTarget` API，Application Auto Scaling 都会检索当前容量并将其与最小容量和最大容量进行比较。如果该容量超出最小-最大范围，则会更新容量以符合设置的最小值和最大值。

横向缩减

当 Application Auto Scaling 自动减少可扩展目标的容量时，可扩展目标将横向缩减。设置扩缩策略后，将无法将可扩展目标横向缩减至最小容量以下。

扩展

当 Application Auto Scaling 自动增加可扩展目标的容量时，可扩展目标将横向扩展。设置扩缩策略后，将无法将可扩展目标横向扩展至最大容量以上。

扩缩策略

扩缩策略指示 Application Auto Scaling 跟踪特定的 CloudWatch 指标。然后，它确定当指标高于或低于某个阈值时要采取的扩缩操作。例如，如果集群中的 CPU 使用率开始上升，您可能希望横向扩展，而当其再次下降时横向缩减。

用于弹性伸缩的指标由目标服务发布，但您也可以将自己的指标发布到 CloudWatch，然后将其与扩缩策略一起使用。

扩缩活动之间的冷却时间可让资源在另一个扩缩活动开始之前稳定下来。Application Auto Scaling 在冷却时间内继续评估指标。冷却时间结束后，扩缩策略将根据需要启动另一个扩缩活动。在冷却时间生效时，如果需要根据当前指标值进行更大的横向扩展，则扩缩策略会立即横向扩展。

计划的操作

计划的操作在特定日期和时间弹性伸缩资源。它们通过修改可扩展目标的最小容量和最大容量来工作，因此可以通过设置更高的最小容量或更低的最大容量用于按计划横向缩减和横向扩展。例如，您可以使用计划的操作来扩缩周末不消耗资源的应用程序，方法是在星期五减少容量，并在下一个星期一增加容量。

您还可以使用计划的操作优化随时间推移的最小值和最大值，以适应预期流量高于正常流量的情况，例如营销活动或季节性波动。这样做可以帮助您在需要横向扩展更高以满足不断增加的使用量时提高性能，并在使用较少的资源时降低成本。

了解更多信息

可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7) - 本节向您介绍可以扩展的服务，并通过注册可扩展目标来帮助您设置弹性伸缩。本节还介绍 Application Auto Scaling 为访问目标服务中的资源而创建的每个 IAM 服务相关角色。

Application Auto Scaling 的目标跟踪扩缩策略 (p. 41) - Application Auto Scaling 的主要功能之一是目标跟踪扩缩策略。了解目标跟踪策略如何根据配置的指标和目标值自动调整所需容量，使利用率保持在恒定水平。例如，您可以配置目标跟踪，使 Spot 实例集的平均 CPU 利用率保持在 50%。然后，Application Auto Scaling 根据需要启动或终止 EC2 实例，以使所有服务器的聚合 CPU 利用率保持在 50%。

可以与 Application Auto Scaling 一起使用的 Amazon 服务

Application Auto Scaling 与其他 Amazon 服务集成，以便您可以添加扩缩功能以满足应用程序的需求。Auto Scaling 是服务的一个可选功能，在几乎所有情况下都默认禁用该功能。

下表列出可以与 Application Auto Scaling 一起使用的 Amazon 服务，包括有关配置弹性伸缩的支持方法的信息。您还可以将 Application Auto Scaling 与自定义资源一起使用。

控制台访问 - 通过在目标服务的控制台中配置扩缩策略，您可以配置兼容的 Amazon 服务以启动弹性伸缩。目前，只有 Amazon AppStream 2.0、ElastiCache 和竞价型实例集为计划的扩缩提供控制台支持。如果服务支持控制台访问，请参阅了解更多信息链接，了解如何从该服务的控制台配置弹性伸缩。

CLI 访问 - 您可以配置兼容的 Amazon 服务，以使用 Amazon CLI 启动弹性伸缩。

软件开发工具包访问 - 您可以配置兼容的 Amazon 服务，以使用 Amazon 软件开发工具包启动弹性伸缩。

CloudFormation 访问 - 您可以配置兼容的 Amazon 服务，以使用 Amazon CloudFormation 堆栈模板启动弹性伸缩。有关更多信息，请参阅使用 Amazon CloudFormation 创建 Application Auto Scaling 资源 (p. 99)。

Amazon 服务	控制台访问	CLI 访问	软件开发工具包访问	CloudFormation 访问
AppStream 2.0 (p. 8)	✓是 了解更多	✓是	✓是	✓是
Aurora (p. 9)	✓是 了解更多	✓是	✓是	✓是
Amazon Comprehend (p. 10)	✗否	✓是	✓是	✓是
Amazon DynamoDB (p. 11)	✓是 了解更多	✓是	✓是	✓是
Amazon ECS (p. 13)	✓是 了解更多	✓是	✓是	✓是
Amazon ElastiCache (p. 14)	✓是 了解更多	✓是	✓是	✓是
Amazon EMR	✓是 了解更多	✓是	✓是	✓是

Amazon 服务	控制台访问	CLI 访问	软件开发工具包访问	CloudFormation 访问
Amazon Keyspaces (p. 15)	✓是 了解更多	✓是	✓是	✓是
Lambda (p. 17)	✗否	✓是	✓是	✓是
Amazon MSK (p. 18)	✓是 了解更多	✓是	✓是	✓是
Amazon Neptune (p. 19)	✗否	✓是	✓是	✓是
SageMaker (p. 20)	✓是 了解更多	✓是	✓是	✓是
竞价型实例集 (p. 21)	✓是 了解更多	✓是	✓是	✓是
自定义资源 (p. 22)	✗否	✓是	✓是	✓是

Amazon AppStream 2.0 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展 AppStream 2.0 队列。

使用以下信息可帮助您将 AppStream 2.0 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 AppStream 2.0 队列，则可以在以下文档中查看有关将 AppStream 2.0 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon AppStream 2.0 管理指南中的适用于 AppStream 2.0 的队列 Auto Scaling

为 AppStream 2.0 创建的服务相关角色

使用 Application Auto Scaling 将 AppStream 2.0 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `appstream.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 AppStream 2.0 队列注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 AppStream 2.0 队列创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 AppStream 2.0 控制台配置弹性伸缩，AppStream 2.0 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 AppStream 2.0 机群调用 register-scalable-target 命令。以下示例注册名为 `sample-fleet` 的队列的所需容量，最小容量为一个队列实例，最大容量为 5 个队列实例。

```
aws application-autoscaling register-scalable-target \
   --service-namespace appstream \
   --scalable-dimension appstream:fleet:DesiredCapacity \
   --resource-id fleet/sample-fleet \
   --min-capacity 1 \
   --max-capacity 5
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon Aurora 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展 Aurora 数据库集群。

使用以下信息可帮助您将 Aurora 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Aurora 数据库集群，则可以在以下文档中查看有关将 Aurora 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon RDS 用户指南中的将 Amazon Aurora Auto Scaling 与 Aurora 副本一起使用

为 Aurora 创建的服务相关角色

使用 Application Auto Scaling 将 Aurora 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_RDSCluster`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `rds.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Aurora 数据库集群注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Aurora 集群创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 Aurora 控制台配置弹性伸缩，Aurora 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Aurora 集群调用 register-scalable-target 命令。以下示例在名为 `my-db-cluster` 的集群中注册 Aurora 副本的计数，最小容量为一个 Aurora 副本，最大容量为 8 个 Aurora 副本。

```
aws application-autoscaling register-scalable-target \
  --service-namespace rds \
  --scalable-dimension rds:cluster:ReadReplicaCount \
  --resource-id cluster:my-db-cluster \
  --min-capacity 1 \
  --max-capacity 8
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon Comprehend 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 Amazon Comprehend 文档分类和实体识别程序终端节点。

使用以下信息可帮助您将 Amazon Comprehend 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Amazon Comprehend 文档分类和实体识别程序终端节点，您可以在以下文档中查看有关将 Amazon Comprehend 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon Comprehend Developer Guide 中的 Auto scaling with endpoints

为 Amazon Comprehend 创建的服务相关角色

使用 Application Auto Scaling 将 Amazon Comprehend 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `comprehend.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Amazon Comprehend 资源注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Amazon Comprehend 文档分类或实体识别程序终端节点创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为文档分类端点调用 register-scalable-target 命令。以下示例使用终端节点的 ARN 注册文档分类程序终端节点模型要使用的所需推理单位数，最小容量为一个推理单位，最大容量为三个推理单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace comprehend \
  --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:document-classifier-
endpoint/EXAMPLE \
  --min-capacity 1 \
  --max-capacity 3
```

  为实体识别程序终端节点调用 `register-scalable-target` 命令。以下示例使用终端节点的 ARN 注册实体识别程序终端节点模型要使用的所需推理单位数，最小容量为一个推理单位，最大容量为三个推理单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace comprehend \
  --scalable-dimension comprehend:entity-recognizer-endpoint:DesiredInferenceUnits \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:entity-recognizer-
endpoint/EXAMPLE \
  --min-capacity 1 \
  --max-capacity 3
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon DynamoDB 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 DynamoDB 表和全局二级索引。

使用以下信息可帮助您将 DynamoDB 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 DynamoDB 表和全局二级索引，您可以在以下文档中查看有关将 DynamoDB 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon DynamoDB 开发人员指南中的使用 DynamoDB Auto Scaling 管理吞吐量

Tip

我们还在 教程：使用 Amazon CLI 的计划扩缩入门 (p. 34) 中提供计划扩缩的教程。在该教程中，您将了解配置扩缩以便您的 DynamoDB 表按计划的时间扩展的基本步骤。

为 DynamoDB 创建的服务相关角色

使用 Application Auto Scaling 将 DynamoDB 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `dynamodb.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 DynamoDB 资源注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 DynamoDB 表或全局二级索引创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 DynamoDB 控制台配置弹性伸缩，DynamoDB 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为表格的写入容量调用 register-scalable-target 命令。以下示例注册名为 `my-table` 的表的预置写入容量，最小容量为 5 个写入容量单位，最大容量为 10 个写入容量单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/my-table \
  --min-capacity 5 \
  --max-capacity 10
```

  为表格的读取容量调用 register-scalable-target 命令。以下示例注册名为 `my-table` 的表的预置读取容量，最小容量为 5 个读取容量单位，最大容量为 10 个读取单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits \
  --resource-id table/my-table \
  --min-capacity 5 \
  --max-capacity 10
```

  为全局二级索引的写入容量调用 register-scalable-target 命令。以下示例注册名为 `my-table-index` 的全局二级索引的预置写入容量，最小容量为 5 个写入容量单位，最大容量为 10 个写入容量单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:index:WriteCapacityUnits \
  --resource-id table/my-table/index/my-table-index \
  --min-capacity 5 \
  --max-capacity 10
```

为全局二级索引的读取容量调用 register-scalable-target 命令。以下示例注册名为 `my-table-index` 的全局二级索引的预置读取容量，最小容量为 5 个读取容量单位，最大容量为 10 个读取容量单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:index:ReadCapacityUnits \
  --resource-id table/my-table/index/my-table-index \
  --min-capacity 5 \
  --max-capacity 10
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon ECS 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展 ECS 服务。

使用以下信息可帮助您将 Amazon ECS 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 ECS 服务，您可以在以下文档中查看有关将 Amazon ECS 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon Elastic Container Service 开发人员指南中的服务 Auto Scaling

为 Amazon ECS 创建的服务相关角色

使用 Application Auto Scaling 将 Amazon ECS 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_ECSService`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `ecs.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 ECS 服务注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Amazon ECS 服务创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 Amazon ECS 控制台配置弹性伸缩，Amazon ECS 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Amazon ECS 服务调用 register-scalable-target 命令。以下示例为名为 `sample-app-service` 的服务（在 `default` 集群上运行）注册可扩展目标，最小任务计数为一个任务，最大任务计数为 10 个任务。

```
aws application-autoscaling register-scalable-target \
  --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/sample-app-service \
  --min-capacity 1 \
  --max-capacity 10
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

ElastiCache for Redis 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 ElastiCache for Redis 复制组。

使用以下信息可帮助您将 ElastiCache 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 ElastiCache for Redis 复制组，则可以在以下文档中查看有关将 ElastiCache 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon ElastiCache for Redis User Guide 中的 Auto Scaling ElastiCache for Redis clusters

为 ElastiCache 创建的服务相关角色

使用 Application Auto Scaling 将 ElastiCache 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `elasticache.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 ElastiCache for Redis 复制组注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 ElastiCache 复制组创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 ElastiCache 控制台配置弹性伸缩，ElastiCache 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 ElastiCache 复制组调用 register-scalable-target 命令。以下示例注册名为 `mycluster` 的复制组的所需节点组数量，最小容量为一个，最大容量为 5 个。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:replication-group:NodeGroups \
   --resource-id replication-group/mycluster \
   --min-capacity 1 \
   --max-capacity 5
```

  以下示例注册名为 `mycluster` 的复制组每个节点组所需的副本数量，最小容量为 1，最大容量为 5。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:replication-group:Replicas \
   --resource-id replication-group/mycluster \
   --min-capacity 1 \
   --max-capacity 5
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon Keyspaces（针对 Apache Cassandra）和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 Amazon Keyspaces 表。

使用以下信息可帮助您将 Amazon Keyspaces 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Amazon Keyspaces 表，您可以在以下文档中查看有关将 Amazon Keyspaces 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon Keyspaces (for Apache Cassandra) Developer Guide 中的 Managing Amazon Keyspaces throughput capacity with Application Auto Scaling

为 Amazon Keyspaces 创建的服务相关角色

使用 Application Auto Scaling 将 Amazon Keyspaces 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_CassandraTable`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `cassandra.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Amazon Keyspaces 表注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Amazon Keyspaces 表创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 Amazon Keyspaces 控制台配置弹性伸缩，Amazon Keyspaces 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Amazon Keyspaces 表调用 register-scalable-target 命令。以下示例注册名为 `mytable` 的表的预置写入容量，最小容量为 5 个写入容量单位，最大容量为 10 个写入容量单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace cassandra \
  --scalable-dimension cassandra:table:WriteCapacityUnits \
  --resource-id keyspace/mykeyspace/table/mytable \
  --min-capacity 5 \
  --max-capacity 10
```

  以下示例注册名为 `mytable` 的表的预置读取容量，最小容量为 5 个读取容量单位，最大容量为 10 个读取容量单位。

```
aws application-autoscaling register-scalable-target \
  --service-namespace cassandra \
  --scalable-dimension cassandra:table:ReadCapacityUnits \
  --resource-id keyspace/mykeyspace/table/mytable \
  --min-capacity 5 \
  --max-capacity 10
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon Lambda 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 Amazon Lambda 预置并发。

使用以下信息可帮助您将 Lambda 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Lambda 函数，则可以在以下文档中查看有关将 Lambda 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon Lambda 开发人员指南中的管理 Lambda 预置并发

为 Lambda 创建的服务相关角色

使用 Application Auto Scaling 将 Lambda 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `lambda.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Lambda 函数注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Lambda 函数创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Lambda 函数调用 register-scalable-target 命令。以下示例为名为 `my-function` 的函数注册别名为 `BLUE` 的预置并发，最小容量为 0，最大容量为 100。

```
aws application-autoscaling register-scalable-target \
  --service-namespace lambda \
  --scalable-dimension lambda:function:ProvisionedConcurrency \
  --resource-id function:my-function:BLUE \
  --min-capacity 0 \
  --max-capacity 100
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon Managed Streaming for Apache Kafka (MSK) 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略横向扩展 Amazon MSK 集群存储。按目标跟踪策略横向缩减已禁用。

使用以下信息可帮助您将 Amazon MSK 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Amazon MSK 集群存储，您可以在以下文档中查看有关将 Amazon MSK 与 Application Auto Scaling 一起使用的详细信息：

- Amazon Managed Streaming for Apache Kafka Developer Guide 中的 Auto-expanding storage for an Amazon MSK cluster

为 Amazon MSK 创建的服务相关角色

使用 Application Auto Scaling 将 Amazon MSK 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_KafkaCluster`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `kafka.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Amazon MSK 集群存储注册为可扩展目标

Application Auto Scaling 需要一个可扩展的目标，然后才能为 Amazon MSK 集群的每个代理的存储卷大小创建扩缩策略。可扩展目标是 Application Auto Scaling 可以扩展的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 Amazon MSK 控制台配置弹性伸缩，Amazon MSK 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Amazon MSK 集群调用 register-scalable-target 命令。以下示例注册 Amazon MSK 集群每个代理的存储卷大小，最小容量为 100 GiB，最大容量为 800 GiB。

```
aws application-autoscaling register-scalable-target \
   --service-namespace kafka \
   --scalable-dimension kafka:broker-storage:VolumeSize \
   --resource-id arn:aws:kafka:us-east-1:123456789012:cluster/demo-
cluster-1/6357e0b2-0e6a-4b86-a0b4-70df934c2e31-5 \
   --min-capacity 100 \
   --max-capacity 800
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

  Note

  当 Amazon MSK 集群是可扩展目标时，横向缩减将禁用且无法启用。

Amazon Neptune 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略和计划的扩缩来扩展 Neptune 集群。

使用以下信息可帮助您将 Neptune 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Neptune 集群，则可以在以下文档中查看有关将 Neptune 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Neptune 用户指南中的自动扩缩 Amazon Neptune 数据库集群中的副本数量

为 Neptune 创建的服务关联角色

使用 Application Auto Scaling 将 Neptune 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `neptune.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Neptune 集群注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Neptune 集群创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Neptune 集群调用 register-scalable-target 命令。以下示例注册名为 `mycluster` 的集群的所需容量，最小容量为一个，最大容量为八个。

```
aws application-autoscaling register-scalable-target \
  --service-namespace neptune \
```

```
--scalable-dimension neptune:cluster:ReadReplicaCount \
--resource-id cluster:mycluster \
--min-capacity 1 \
--max-capacity 8
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon SageMaker 和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展 SageMaker 终端节点变体。

使用以下信息可帮助您将 SageMaker 与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 SageMaker 终端节点变体，则可以在以下文档中查看有关将 SageMaker 与 Application Auto Scaling 一起使用的示例配置和详细信息：

- Amazon SageMaker Developer Guide 中的 Automatically scale Amazon SageMaker models

为 SageMaker 创建的服务相关角色

使用 Application Auto Scaling 将 SageMaker 资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `sagemaker.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 SageMaker 终端节点变体注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 SageMaker xxx 创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 SageMaker 控制台配置弹性伸缩，SageMaker 会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 SageMaker 端点变体调用 register-scalable-target 命令。以下示例注册名为 `my-variant` 的产品变体（在 `my-endpoint` 终端节点上运行）所需的 EC2 实例计数，最小容量为一个实例，最大容量为 8 个实例。

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --scalable-dimension sagemaker:variant:DesiredInstanceCount \
  --resource-id endpoint/my-endpoint/variant/my-variant \
  --min-capacity 1 \
  --max-capacity 8
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

Amazon EC2 Spot 实例集和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展 Spot 实例集。

使用以下信息可帮助您将 Spot 实例集与 Application Auto Scaling 集成。

如果您刚刚开始扩缩 Spot 实例集，则可以在以下文档中查看有关将 Spot 实例集与 Application Auto Scaling 一起使用的详细信息：

- Amazon EC2 用户指南中的 Spot 实例集的自动扩展

为 Spot 实例集创建的服务相关角色

使用 Application Auto Scaling 将 Spot 实例集资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `ec2.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 Spot 实例集注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为 Spot 实例集创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

如果您使用 Spot 实例集控制台配置弹性伸缩，Spot 实例集会自动为您注册一个可扩展的目标。

如果要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

  为 Spot 实例集调用 register-scalable-target 命令。以下示例使用其请求 ID 注册 Spot 实例集的目标容量，最小容量为两个实例，最大容量为 10 个实例。

```
aws application-autoscaling register-scalable-target \
  --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --min-capacity 2 \
  --max-capacity 10
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 `ResourceId`、`ScalableDimension`、`ServiceNamespace`、`MinCapacity` 和 `MaxCapacity` 作为参数。

自定义资源和 Application Auto Scaling

您可以使用目标跟踪扩缩策略、分步扩缩策略和计划的扩缩来扩展自定义资源。

使用以下信息可帮助您将自定义资源与 Application Auto Scaling 集成。

如果您刚刚开始扩缩自定义资源，您可以查看我们的 GitHub 存储库，其中提供了有关如何将自定义资源与 Application Auto Scaling 集成的详细信息。

为自定义资源创建服务相关角色

使用 Application Auto Scaling 将自定义资源注册为可扩展目标时，将在您的 Amazon Web Services 账户 中自动创建以下服务相关角色。此角色允许 Application Auto Scaling 在您的账户中执行受支持的操作。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

- `AWSServiceRoleForApplicationAutoScaling_CustomResource`

服务相关角色使用的服务委托人

上一节中的服务相关角色只能由为角色定义的信任关系授权的服务委托人担任。Application Auto Scaling 使用的服务相关角色为以下服务委托人授予访问权限：

- `custom-resource.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将自定义资源注册为可扩展目标

Application Auto Scaling 需要一个可扩展目标，然后才能为自定义资源创建扩缩策略或计划的操作。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。可扩展目标由资源 ID、可扩展维度和命名空间的组合唯一标识。

要使用 Amazon CLI 或 Amazon 软件开发工具包之一配置弹性伸缩，您可以使用以下选项：

- Amazon CLI:

为自定义资源调用 register-scalable-target 命令。以下示例将自定义资源注册为可扩展目标，最小所需计数为一个容量单位，最大所需计数为 10 个容量单位。custom-resource-id.txt 文件包含一个标识资源 ID 的字符串，它表示通过 Amazon API Gateway 终端节点到自定义资源的路径。

```
aws application-autoscaling register-scalable-target \
  --service-namespace custom-resource \
  --scalable-dimension custom-resource:ResourceType:Property \
  --resource-id file://~/custom-resource-id.txt \
  --min-capacity 1 \
  --max-capacity 10
```

custom-resource-id.txt 的内容：

```
https://example.execute-api.us-west-2.amazonaws.com/prod/
scalableTargetDimensions/1-23456789
```

- Amazon 软件开发工具包：

  调用 RegisterScalableTarget 操作并提供 ResourceId、ScalableDimension、ServiceNamespace、MinCapacity 和 MaxCapacity 作为参数。

Application Auto Scaling 的计划扩缩

根据计划扩缩使您可以按照可预测的负载变化来设置您自己的扩缩计划。例如，假设您的 Web 应用程序的流量会在每周的星期三开始增加，并在星期四保持高流量状态，然后在星期五开始下降。您可以配置 Application Auto Scaling 的计划，以在星期三增加容量并在星期五减少容量。

要使用计划的扩缩，请创建指示 Application Auto Scaling 在特定时间执行扩缩活动的计划操作。创建计划的操作时，请指定可扩展目标、应进行扩缩活动的时间以及最小和最大容量。您可以创建仅扩展一次或按重复计划扩展的计划操作。

在指定的时间，Application Auto Scaling 通过将当前容量与指定的最小容量和最大容量进行比较，根据新容量值进行扩展。

- 如果当前容量小于指定的最小容量，Application Auto Scaling 将横向扩展（增加容量）到指定的最小容量。
- 如果当前容量大于指定的最大容量，Application Auto Scaling 将横向缩减（减少容量）到指定的最大容量。

您可以在同一资源上同时使用计划的扩缩和扩缩策略，以获得两者的好处。运行计划的操作后，扩缩策略可以继续决定是否进一步扩展容量。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。

有关使用计划扩缩的详细示例，请参阅 Amazon 计算博客上的博客文章计划 Amazon Lambda 预置并发性以实现重复使用峰值。有关演练如何使用示例 Amazon 资源创建计划操作的教程，请参阅 教程：使用 Amazon CLI 的计划扩缩入门 (p. 34)。

注意事项

创建计划的操作时，请记住以下内容：

- 计划操作将 `MinCapacity` 和 `MaxCapacity` 设置为由计划操作在指定的日期和时间指定的内容。请求可以选择只包含这些大小中的一个。例如，您可以创建仅指定最小容量的计划操作。但是，在某些情况下，您必须包括两种大小，以确保新的最小容量不大于最大容量，或者新的最大容量不小于最小容量。
- 预设情况下，您设置的重复计划采用协调世界时 (UTC)。您可以更改时间以符合本地时区或您的网络中其他部分的时区。如果您指定的时区遵守夏令时，则操作会自动调整夏令时 (DST)。有关更多信息，请参阅使用 cron 表达式安排重复发生的扩缩操作 (p. 25)。
- 您可以临时关闭可扩展目标的计划扩缩。这有助于防止计划操作处于活动状态，而无需将其删除。然后，当您想要再次使用时，您可以恢复计划的扩展。有关更多信息，请参阅暂停和恢复 Application Auto Scaling 的扩缩 (p. 63)。
- 将会保证同一可扩展目标的计划操作运行顺序，但不保证不同可扩展目标中的计划操作的执行顺序。
- 要成功完成计划操作，目标服务中的指定资源必须位于可扩展状态。如果不是此状态，则请求将会失败，并返回错误消息，例如：`Resource Id [ActualResourceId] is not scalable. Reason: The status of all DB instances must be 'available' or 'incompatible-parameters'`。
- 由于 Application Auto Scaling 和目标服务的分布式特性，计划操作触发时间与目标服务实际执行扩缩操作的时间之间的延迟可能有几秒钟。因为系统将按照指定操作的顺序来运行计划的操作，所以开始时间彼此接近的计划操作可能需要更长时间才能运行。

计划操作创建、管理和删除的常用命令

使用计划扩缩的常用命令包括：

- register-scalable-target 将 Amazon 或自定义资源注册为可扩展目标（Application Auto Scaling 可以扩展的资源），并暂停和恢复扩缩。
- put-scheduled-action 可添加或修改现有可扩展目标的计划操作。
- describe-scaling-activities 可返回关于 Amazon 区域中扩缩活动的信息。
- describe-scheduled-actions 可返回关于 Amazon 区域中计划操作的信息。
- delete-scheduled-action 可删除计划的操作。

限制

以下是使用计划的扩缩时的限制：

- 每个可扩展目标的计划操作的名称必须是唯一的。
- Application Auto Scaling 不在计划表达式中提供二级精度。使用 Cron 表达式的最高解析精度是一分钟。
- 可扩展目标不能是 Amazon MSK 集群。Amazon MSK 不支持计划的扩缩。

使用 cron 表达式安排重复发生的扩缩操作

您可以使用 cron 表达式创建定期运行的计划操作。

若要创建重复计划，请指定 cron 表达式和时区来描述何时重复执行该计划操作。支持的时区值为 Joda-Time支持的 IANA 时区的规范名（例如 `Etc/GMT+9` 或 `Pacific/Tahiti`）。您可以选择指定开始时间和/或结束时间的日期和时间。有关使用 Amazon CLI 创建计划操作的示例命令，请参阅 创建指定时区的重复计划操作 (p. 29)。

受支持的 cron 表达式格式由用空格分隔的六个字段组成：[Minutes] [Hours] [Day_of_Month] [Month] [Day_of_Week] [Year]。例如，Cron 表达式 `30 6 * * 2 *` 配置每周二的早上 6:30 再执行的计划操作。星号用作通配符，以匹配字段的所有值。有关编写 cron 表达式的更多信息，请参阅 Amazon CloudWatch Events 用户指南中的 Cron 表达式。

当您创建重复性计划时，请谨慎选择开始时间和结束时间。记住以下内容：

- 如果您指定开始时间，则 Application Auto Scaling 将在此时间执行操作，然后根据指定的重复执行操作。
- 如果指定结束时间，则操作在此时间之后停止重复。Application Auto Scaling 不会一直跟踪以前的值，并在结束时间后恢复为以前的值。
- 使用 Amazon CLI 或 Amazon SDK 创建或更新计划操作时，开始时间和结束时间必须设置为 UTC 时间。

示例

为 Application Auto Scaling 可扩展目标创建重复性计划时，可您可以参考下表。以下示例展示了使用 Application Auto Scaling 创建或更新计划操作的正确语法。

分钟	小时	日期	月份	星期几	年份	意义
0	10	*	*	?	*	每天上午的 10:00 (UTC) 运行

分钟	小时	日期	月份	星期几	年份	意义
15	12	*	*	?	*	每天在下午 12:15 (UTC) 运行
0	18	?	*	MON-FRI	*	每星期一到星期五的下午 6:00 (UTC) 运行
0	8	1	*	?	*	每月第 1 天的上午 8:00 (UTC) 运行
0/15	*	*	*	?	*	每 15 分钟运行一次
0/10	*	?	*	MON-FRI	*	从星期一到星期五，每 10 分钟运行一次
0/5	8-17	?	*	MON-FRI	*	每星期一到星期五的上午 8:00 和下午 5:55 (UTC) 之间，每 5 分钟运行一次

例外

此外，您还可以使用包含七个字段的字符串值创建 cron 表达式。在这种情况下，您可以使用前三个字段来指定运行计划操作的时间，包括秒数。完整的 cron 表达式包含以下以空格分隔的字段：[Seconds] [Minutes] [Hours] [Day_of_Month] [Month] [Day_of_Week] [Year]。但是，这种方法不能保证计划操作会在您指定的准确秒数运行。此外，某些服务控制台可能不支持 cron 表达式中的秒数字段。

Application Auto Scaling 的计划操作示例

以下示例展示如何使用 Amazon CLI put-scheduled-action 命令创建计划的操作。当您指定新容量时，可指定最小容量和/或最大容量。

> **Note**
>
> 为简洁起见，本主题中的示例说明与 Application Auto Scaling 集成的一些服务的 CLI 命令。要指定不同的可扩展目标，请在 `--service-namespace` 中指定其命名空间，在 `--scalable-dimension` 中指定其可扩展维度，并在 `--resource-id` 中指定其资源 ID。

目录

- 创建指定时区的重复计划操作 (p. 29)

创建仅发生一次的计划操作

要在指定的日期和时间仅弹性伸缩可扩展目标一次，请使用 `--schedule "at(yyyy-mm-ddThh:mm:ss)"` 选项。

Example 示例：仅向外扩展一次

以下是创建计划操作以在特定日期和时间横向扩展容量的示例。

在为 `--schedule` 指定的日期和时间（UTC 时间 2021 年 3 月 31 日晚上 10:00），如果为 `MinCapacity` 指定的值高于当前容量，则 Application Auto Scaling 将横向扩展到 `MinCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource \
  --scalable-dimension custom-resource:ResourceType:Property \
  --resource-id file://~/custom-resource-id.txt \
  --scheduled-action-name scale-out \
  --schedule "at(2021-03-31T22:00:00)" \
  --scalable-target-action MinCapacity=3
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource --
scalable-dimension custom-resource:ResourceType:Property --resource-id file://~/custom-
resource-id.txt --scheduled-action-name scale-out --schedule "at(2021-03-31T22:00:00)" --
scalable-target-action MinCapacity=3
```

> **Note**
>
> 运行此计划操作时，如果最大容量小于为最小容量指定的值，则必须指定新的最小容量和最大容量，而不仅仅是最小容量。

Example 示例：仅向内扩展一次

以下是创建计划操作以在特定日期和时间横向缩减容量的示例。

在为 `--schedule` 指定的日期和时间（UTC 时间 2021 年 3 月 31 日晚上 10:30），如果为 `MaxCapacity` 指定的值低于当前容量，则 Application Auto Scaling 将横向缩减到 `MaxCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource \
  --scalable-dimension custom-resource:ResourceType:Property \
  --resource-id file://~/custom-resource-id.txt \
  --scheduled-action-name scale-in \
  --schedule "at(2021-03-31T22:30:00)" \
  --scalable-target-action MinCapacity=0,MaxCapacity=0
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace custom-resource --
scalable-dimension custom-resource:ResourceType:Property --resource-id file://~/custom-
```

```
resource-id.txt --scheduled-action-name scale-in --schedule "at(2021-03-31T22:30:00)" --
scalable-target-action MinCapacity=0,MaxCapacity=0
```

创建按重复间隔运行的计划操作

要按重复间隔计划扩缩，请使用 `--schedule "rate(value unit)"` 选项。该值必须为正整数。单位可以是 `minute`、`minutes`、`hour`、`hours`、`day` 或 `days`。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的 Rate 表达式。

以下是使用 rate 表达式的计划操作的示例。

根据指定的计划（从 UTC 时间 2021 年 1 月 30 日中午 12:00 PM 开始，到 UTC 时间 2021 年 1 月 31 日晚上 10:00 结束，每 5 个小时一次），如果为 `MinCapacity` 指定的值高于当前容量，则 Application Auto Scaling 横向扩展到 `MinCapacity`。如果为 `MaxCapacity` 指定的值低于当前容量，则 Application Auto Scaling 将横向缩减到 `MaxCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/web-app \
  --scheduled-action-name my-recurring-action \
  --schedule "rate(5 hours)" \
  --start-time 2021-01-30T12:00:00 \
  --end-time 2021-01-31T22:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace ecs --scalable-
dimension ecs:service:DesiredCount --resource-id service/default/web-app --scheduled-
action-name my-recurring-action --schedule "rate(5 hours)" --start-time 2021-01-30T12:00:00
 --end-time 2021-01-31T22:00:00 --scalable-target-action MinCapacity=3,MaxCapacity=10
```

创建按重复计划运行的计划操作

要按重复计划来计划扩缩，请使用 `--schedule "cron(fields)"` 选项。有关更多信息，请参阅使用 cron 表达式安排重复发生的扩缩操作 (p. 25)。

以下是使用 cron 表达式的计划操作的示例。

根据指定的计划（UTC 时间每天上午 9:00），如果为 `MinCapacity` 指定的值高于当前容量，则 Application Auto Scaling 将横向扩展到 `MinCapacity`。如果为 `MaxCapacity` 指定的值低于当前容量，则 Application Auto Scaling 将横向缩减到 `MaxCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace appstream \
  --scalable-dimension appstream:fleet:DesiredCapacity \
  --resource-id fleet/sample-fleet \
  --scheduled-action-name my-recurring-action \
  --schedule "cron(0 9 * * ? *)" \
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace appstream --scalable-
dimension appstream:fleet:DesiredCapacity --resource-id fleet/sample-fleet --scheduled-
```

```
action-name my-recurring-action --schedule "cron(0 9 * * ? *)" --scalable-target-action
 MinCapacity=10,MaxCapacity=50
```

创建指定时区的一次性计划操作

默认情况下，计划操作设置为 UTC 时区。要指定不同的时区，请包含 `--timezone` 选项并指定时区的规范名称（例如，`America/New_York`）。有关更多信息，请参阅 https://www.joda.org/joda-time/timezones.html，该网站提供关于调用 put-scheduled-action 时支持的 IANA 时区的信息。

以下是创建计划操作以在特定日期和时间扩展容量时使用 `--timezone` 选项的示例。

在为 `--schedule` 指定的日期和时间（当地时间 2021 年 1 月 31 日下午 5:00），如果为 `MinCapacity` 指定的值高于当前容量，则 Application Auto Scaling 将横向扩展到 `MinCapacity`。如果为 `MaxCapacity` 指定的值低于当前容量，则 Application Auto Scaling 将横向缩减到 `MaxCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace comprehend \
  --scalable-dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits \
  --resource-id arn:aws:comprehend:us-west-2:123456789012:document-classifier-endpoint/
EXAMPLE \
  --scheduled-action-name  my-one-time-action \
  --schedule "at(2021-01-31T17:00:00)" --timezone "America/New_York" \
  --scalable-target-action MinCapacity=1,MaxCapacity=3
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace comprehend --scalable-
dimension comprehend:document-classifier-endpoint:DesiredInferenceUnits --resource-
id arn:aws:comprehend:us-west-2:123456789012:document-classifier-endpoint/EXAMPLE --
scheduled-action-name  my-one-time-action --schedule "at(2021-01-31T17:00:00)" --timezone
 "America/New_York" --scalable-target-action MinCapacity=1,MaxCapacity=3
```

创建指定时区的重复计划操作

以下是创建重复计划操作以扩展容量时使用 `--timezone` 选项的示例。有关更多信息，请参阅使用 cron 表达式安排重复发生的扩缩操作 (p. 25)。

根据指定的计划（当地时间每个星期一到星期五晚上 6:00），如果为 `MinCapacity` 指定的值高于当前容量，则 Application Auto Scaling 将横向扩展到 `MinCapacity`。如果为 `MaxCapacity` 指定的值低于当前容量，则 Application Auto Scaling 将横向缩减到 `MaxCapacity`。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action --service-namespace lambda \
  --scalable-dimension lambda:function:ProvisionedConcurrency \
  --resource-id function:my-function:BLUE \
  --scheduled-action-name my-recurring-action \
  --schedule "cron(0 18 ? * MON-FRI *)" --timezone "Etc/GMT+9" \
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace lambda --scalable-
dimension lambda:function:ProvisionedConcurrency --resource-id function:my-function:BLUE --
scheduled-action-name my-recurring-action --schedule "cron(0 18 ? * MON-FRI *)" --timezone
 "Etc/GMT+9" --scalable-target-action MinCapacity=10,MaxCapacity=50
```

管理 Application Auto Scaling 的计划扩缩

Amazon CLI 包括其他几个可帮助您管理计划操作的命令。

> **Note**
>
> 为简洁起见，本主题中的示例说明与 Application Auto Scaling 集成的一些服务的 CLI 命令。要指定不同的可扩展目标，请在 `--service-namespace` 中指定其命名空间，在 `--scalable-dimension` 中指定其可扩展维度，并在 `--resource-id` 中指定其资源 ID。

目录

查看指定服务的扩缩活动

要查看指定服务命名空间中所有可扩展目标的扩缩活动，请使用 describe-scaling-activities 命令。

以下示例检索与 `dynamodb` 服务命名空间关联的扩缩活动。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-activities --service-namespace dynamodb
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace dynamodb
```

如果命令成功，则将显示类似于以下内容的输出。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/my-table",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
            "ResourceId": "table/my-table",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-second-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity to 10",
```

```
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/my-table",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/my-table",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max capacity to 20",
            "StatusCode": "Successful"
        }
    ]
}
```

要更改此命令以使其仅检索其中一个可扩展目标的扩缩活动，请添加 `--resource-id` 选项。

描述指定服务的所有计划操作

要描述指定服务命名空间中所有可扩展目标的计划操作，请使用 describe-scheduled-actions 命令。

以下示例检索与 `ec2` 服务命名空间关联的计划操作。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2
```

如果成功，该命令返回类似以下内容的输出。

```
{
    "ScheduledActions": [
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:493a6261-fbb9-432d-855d-3c302c14bdb9:resource/ec2/spot-
fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE:scheduledActionName/my-one-time-
action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2021-01-31T17:00:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MaxCapacity": 1
```

```
            },
            "CreationTime": 1607454792.331
        },
        {
            "ScheduledActionName": "my-recurring-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:493a6261-fbb9-432d-855d-3c302c14bdb9:resource/ec2/spot-
fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE:scheduledActionName/my-recurring-
action",
            "ServiceNamespace": "ec2",
            "Schedule": "rate(5 minutes)",
            "ResourceId": "spot-fleet-request/sfr-107dc873-0802-4402-a901-37294EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "StartTime": 1604059200.0,
            "EndTime": 1612130400.0,
            "ScalableTargetAction": {
                "MinCapacity": 3,
                "MaxCapacity": 10
            },
            "CreationTime": 1607454949.719
        },
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:4bce34c7-bb81-4ecf-b776-5c726efb1567:resource/ec2/spot-
fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE:scheduledActionName/my-one-time-
action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2020-12-08T9:36:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MinCapacity": 1,
                "MaxCapacity": 3
            },
            "CreationTime": 1607456031.391
        }
    ]
}
```

描述可扩展目标的一个或多个计划操作

要检索有关指定可扩展目标的计划操作的信息，请在使用 describe-scheduled-actions 命令描述计划操作时添加 `--resource-id` 选项。

如果您包含 `--scheduled-action-names` 选项并将计划操作的名称指定为其值，则该命令仅返回名称匹配的计划操作，如以下示例所示。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2 \
  --resource-id spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE \
  --scheduled-action-names my-one-time-action
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace ec2 --resource-
id spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE --scheduled-action-names my-
one-time-action
```

下面是示例输出。

```
{
    "ScheduledActions": [
        {
            "ScheduledActionName": "my-one-time-action",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:4bce34c7-bb81-4ecf-b776-5c726efb1567:resource/ec2/spot-
fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE:scheduledActionName/my-one-time-
action",
            "ServiceNamespace": "ec2",
            "Schedule": "at(2020-12-08T9:36:00)",
            "Timezone": "America/New_York",
            "ResourceId": "spot-fleet-request/sfr-40edeb7b-9ae7-44be-bef2-5c4c8EXAMPLE",
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "ScalableTargetAction": {
                "MinCapacity": 1,
                "MaxCapacity": 3
            },
            "CreationTime": 1607456031.391
        }
    ]
}
```

如果为 `--scheduled-action-names` 选项提供多个值，则名称匹配的所有计划操作都包含在输出中。

关闭可扩展目标的计划扩缩

您可以暂时关闭计划扩缩而不删除您的计划操作。有关更多信息，请参阅暂停和恢复 Application Auto Scaling 的扩缩 (p. 63)。

通过使用 register-scalable-target 命令及 `--suspended-state` 选项，并指定 `true` 作为 `ScheduledScalingSuspended` 属性的值，暂停可扩展目标的计划扩缩，如以下示例所示。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace rds \
  --scalable-dimension rds:cluster:ReadReplicaCount --resource-id cluster:my-db-cluster \
  --suspended-state '{"ScheduledScalingSuspended": true}'
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace rds --scalable-
dimension rds:cluster:ReadReplicaCount --resource-id cluster:my-db-cluster --suspended-
state "{\"ScheduledScalingSuspended\": true}"
```

如果成功，该命令将返回到提示符。

要恢复计划扩缩，请再次运行此命令，指定 `false` 作为 `ScheduledScalingSuspended` 属性的值。

删除计划的操作

完成计划的操作后，您可以使用 delete-scheduled-action 命令将其删除。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scheduled-action --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE \
  --scheduled-action-name my-recurring-action
```

Windows

```
aws application-autoscaling delete-scheduled-action --service-namespace ec2 --scalable-
dimension ec2:spot-fleet-request:TargetCapacity --resource-id spot-fleet-request/
sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE --scheduled-action-name my-recurring-action
```

如果成功，该命令将返回到提示符。

教程：使用 Amazon CLI 的计划扩缩入门

以下教程向您介绍如何使用 Amazon CLI 开始使用计划扩缩，方法是帮助您创建用于扩展名为 `TestTable` 的示例 DynamoDB 表的计划操作。如果您的 DynamoDB 中没有用于测试的 `TestTable` 表，则可以通过运行 Amazon DynamoDB 开发人员指南中的步骤 1：创建 DynamoDB 表中所示的 create-table 命令来立即创建 一个。

当使用 Amazon CLI 时，请记住您的命令在为您的配置文件配置的 Amazon 区域中运行。如果您想要在不同的区域中运行命令，可以为配置文件更改默认区域，或者与命令一起使用 `--region` 参数。

> **Note**
>
> 使用本教程的过程中，您可能会产生 Amazon 费用。请监控免费套餐使用情况，并确保您了解与 DynamoDB 数据库使用的读取和写入容量单位数关联的成本。

目录

步骤 1：注册您的可扩展目标

首先使用 Application Auto Scaling 将您的 DynamoDB 表注册为可扩展目标。

向 Application Auto Scaling 注册您的可扩展目标

1. 首先，请使用 describe-scalable-targets 命令来检查是否已注册任何 DynamoDB 资源。这可以让您验证 `TestTable` 表是否已取消注册，以防它不是新表。

   Linux、macOS 或 Unix

   ```
   aws application-autoscaling describe-scalable-targets \
     --service-namespace dynamodb
   ```

   Windows

   ```
   aws application-autoscaling describe-scalable-targets --service-namespace dynamodb
   ```

   如果没有现有的可扩展目标，则这是系统的响应。

   ```
   {
       "ScalableTargets": []
   }
   ```

2. 使用以下 register-scalable-target 命令来注册名为 `TestTable` 的 DynamoDB 表的写入容量。将最小所需容量设置为 5 个写入容量单位，将最大所需容量设置为 10 个写入容量单位。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/TestTable \
  --min-capacity 5 --max-capacity 10
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable --
min-capacity 5 --max-capacity 10
```

如果此命令成功执行，将不会返回任何输出。

步骤 2：创建两个计划操作

Application Auto Scaling 可让您安排扩缩操作应发生的时间。您可以指定可扩展目标、扩展计划、最小容量和最大容量。在指定的时间，Application Auto Scaling 会更新可扩展目标的最小值和最大值。如果当前容量超出此范围，这会导致一个扩展活动。

如果您决定创建扩展策略，计划更新最小和最大容量也会有所帮助。扩展策略允许基于当前资源利用率来动态扩展您的资源。扩展策略的一种常见保护措施是设置适当的最小和最大容量值。

在本练习中，我们将创建两个一次性操作来分别进行扩展和缩减。

创建和查看计划操作

1. 要创建第一个计划操作，请使用以下 put-scheduled-action 命令。

   `--schedule` 中的 at 命令计划在将来的指定日期和时间要运行一次的操作。小时采用世界标准时间 24 小时格式。将操作安排在当前时间开始约 5 分钟后发生。

   在指定的日期和时间，Application Auto Scaling 将更新 `MinCapacity` 和 `MaxCapacity` 的值。假设表当前有 5 个写入容量单位，Application Auto Scaling 横向扩展到 `MinCapacity`，以使表拥有 15-20 个写入容量单位的新所需范围。

   Linux、macOS 或 Unix

   ```
   aws application-autoscaling put-scheduled-action \
     --service-namespace dynamodb \
     --scalable-dimension dynamodb:table:WriteCapacityUnits \
     --resource-id table/TestTable \
     --scheduled-action-name my-first-scheduled-action \
     --schedule "at(2019-05-20T17:05:00)" \
     --scalable-target-action MinCapacity=15,MaxCapacity=20
   ```

   Windows

   ```
   aws application-autoscaling put-scheduled-action --service-namespace dynamodb --
   scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable --
   scheduled-action-name my-first-scheduled-action --schedule "at(2019-05-20T17:05:00)" --
   scalable-target-action MinCapacity=15,MaxCapacity=20
   ```

   如果此命令成功执行，将不会返回任何输出。

2. 要创建 Application Auto Scaling 用于横向缩减的第二个计划操作，请使用以下 put-scheduled-action 命令。

   将操作安排在当前时间开始约 10 分钟后发生。

   在指定的日期和时间，Application Auto Scaling 将更新表的 `MinCapacity` 和 `MaxCapacity`，并横向缩减到 `MaxCapacity` 以将表恢复为 5-10 个写入容量单位的初始所需范围。

   Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/TestTable \
  --scheduled-action-name my-second-scheduled-action \
  --schedule "at(2019-05-20T17:10:00)" \
  --scalable-target-action MinCapacity=5,MaxCapacity=10
```

   Windows

```
aws application-autoscaling put-scheduled-action --service-namespace dynamodb --
scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable --
scheduled-action-name my-second-scheduled-action --schedule "at(2019-05-20T17:10:00)"
 --scalable-target-action MinCapacity=5,MaxCapacity=10
```

3. （可选）您可以使用以下 describe-scheduled-actions 命令获得指定服务命名空间的计划操作列表。

   Linux、macOS 或 Unix

```
aws application-autoscaling describe-scheduled-actions \
  --service-namespace dynamodb
```

   Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace dynamodb
```

   下面是示例输出。

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/TestTable",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/
dynamodb/table/TestTable:scheduledActionName/my-first-scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 15,
                "MaxCapacity": 20
            },
            "ScheduledActionName": "my-first-scheduled-action",
            "ServiceNamespace": "dynamodb"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:40:00)",
            "ResourceId": "table/TestTable",
            "CreationTime": 1561571946.021,
```

```
            "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:123456789012:scheduledAction:2d36aa3b-cdf9-4565-b290-81db519b227d:resource/
dynamodb/table/TestTable:scheduledActionName/my-second-scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 5,
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-second-scheduled-action",
            "ServiceNamespace": "dynamodb"
        }
    ]
}
```

步骤 3：查看扩缩活动

在此步骤中，您将查看计划操作触发的扩缩活动，并验证 DynamoDB 是否已更改表的写入容量。

查看扩展活动

1. 等待您选择的时间，使用以下 describe-scaling-activities 命令确认计划操作在正常运行。

   Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-activities \
  --service-namespace dynamodb
```

   Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace dynamodb
```

   以下是第一个计划操作在计划操作正在执行时的示例输出。

   扩展活动按创建日期排序，首先返回最新的扩展活动。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/TestTable",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Waiting for
 change to be fulfilled by dynamodb.",
            "StatusCode": "InProgress"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/TestTable",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max capacity to
 20",
            "StatusCode": "Successful"
```

```
        }
    ]
}
```

以下是两个计划操作都运行完成后的示例输出。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/TestTable",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
            "ResourceId": "table/TestTable",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-second-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity to
 10",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/TestTable",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
 successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/TestTable",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max capacity to
 20",
            "StatusCode": "Successful"
        }
    ]
}
```

2. 成功运行计划操作后，请打开 DynamoDB 控制台并选择要处理的表。查看 Capacity（容量）选项卡下的 Write capacity units（写入容量单位）。在第二个扩展操作运行后，写入容量单位应已从 15 变为 10。

此外，您还可以使用以下 describe-table 命令验证表的当前写入容量。包含 `--query` 选项以筛选输出。有关 Amazon CLI 的输出筛选功能的更多信息，请参阅 Amazon Command Line Interface 用户指南中的控制 Amazon CLI 的命令输出。

Linux、macOS 或 Unix

```
aws dynamodb describe-table --table-name TestTable \
  --query 'Table.[TableName,TableStatus,ProvisionedThroughput]'
```

Windows

```
aws dynamodb describe-table --table-name TestTable --query "Table.
[TableName,TableStatus,ProvisionedThroughput]"
```

下面是示例输出。

```
[
    "TestTable",
    "ACTIVE",
    {
        "NumberOfDecreasesToday": 1,
        "WriteCapacityUnits": 10,
        "LastIncreaseDateTime": 1561574133.264,
        "ReadCapacityUnits": 5,
        "LastDecreaseDateTime": 1561574435.607
    }
]
```

步骤 4：后续步骤

如果您想尝试同时使用计划扩展和扩展策略进行扩展，请按照中的步骤操作教程：配置弹性伸缩以处理繁重的工作负载 (p. 56)。

第 5 步：清除

当您完成入门练习后，可以按照如下步骤清除关联的资源。

删除计划的操作

以下 delete-scheduled-action 命令可删除指定的计划操作。如果您要将此计划操作保留供将来使用，您可以跳过此操作。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scheduled-action \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/TestTable \
  --scheduled-action-name my-second-scheduled-action
```

Windows

```
aws application-autoscaling delete-scheduled-action --service-namespace dynamodb --
scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable --
scheduled-action-name my-second-scheduled-action
```

撤消可扩展目标的注册

使用以下 deregister-scalable-target 命令可取消注册可扩展目标。如果您有任何您创建的扩展策略或尚未删除的计划操作，这条命令会将它们删除。如果您要将此可扩展目标保留供将来使用，您可以跳过此操作。

Linux、macOS 或 Unix

```
aws application-autoscaling deregister-scalable-target \
  --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/TestTable
```

Windows

```
aws application-autoscaling deregister-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:WriteCapacityUnits --resource-id table/TestTable
```

删除 DynamoDB 表

使用以下 delete-table 命令可删除本教程中使用的表。如果您要保留该表供将来使用，可以跳过此步骤。

Linux、macOS 或 Unix

```
aws dynamodb delete-table --table-name TestTable
```

Windows

```
aws dynamodb delete-table --table-name TestTable
```

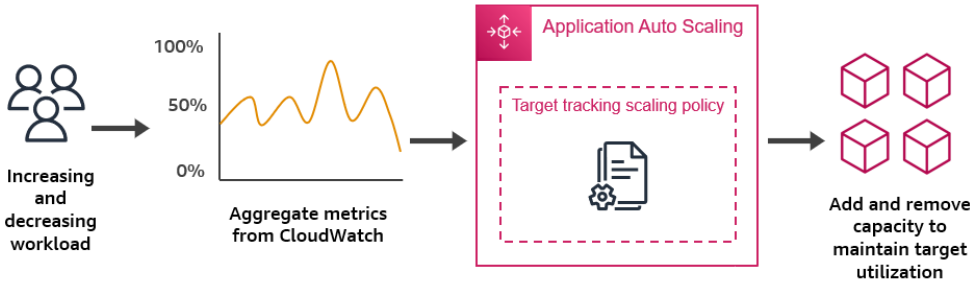
Application Auto Scaling 的目标跟踪扩缩策略

要创建目标跟踪扩缩策略，您需要指定一个 Amazon CloudWatch 指标和一个目标值，目标值用来表示应用程序的理想平均利用率或吞吐量水平。然后 Application Auto Scaling 可以横向扩展可扩展目标（增加容量）以处理峰值流量，并在利用率或吞吐量较低时横向缩减该可扩展目标（删除容量）以降低成本。

例如，假设您当前有一个在竞价型实例集上运行的应用程序，并希望在应用程序负载变化时将该实例集的 CPU 利用率保持在 50% 左右。这为您提供额外容量以处理流量高峰，而无需维护过多的空闲资源。

创建一个将目标平均 CPU 利用率设置为 50% 的目标跟踪扩缩策略即可满足此需求。然后，Application Auto Scaling 会根据需要增减实例数量，以确保实际指标值等于或接近 50%。

下图概述显示设置完成时目标跟踪扩缩策略的工作原理。



注意事项

使用目标跟踪扩缩策略时，需要注意以下事项：

- 请勿创建、编辑或删除与目标跟踪扩缩策略一起使用的 CloudWatch 警报。Application Auto Scaling 会创建并管理与您的目标跟踪扩缩策略关联的 CloudWatch 警报，并在不需要时将其删除。
- 为了确保应用程序可用性，Application Auto Scaling 针对指标尽快按比例横向扩展，但会逐渐横向缩减。有关更多信息，请参阅在高利用率期间支持应用程序可用性 (p. 42)。
- 如果指标缺少数据点，则会导致 CloudWatch 警报状态变为 `INSUFFICIENT_DATA`。发生这种情况时，在找到新的数据点之前，Application Auto Scaling 无法扩展您的可扩展目标。有关在数据不足时创建警报的信息，请参阅 使用 CloudWatch 警报进行监控 (p. 67)。
- 您可能会看到目标值与实际指标数据点之间存在差距。这是因为 Application Auto Scaling 在确定要添加或删除多少容量时将始终通过向上或向下舍入保守地进行操作，以免添加的容量不足或删除的容量过多。但是，对于具有小容量的可扩展目标，实际指标数据点可能看起来与目标值差距很大。
- 对于容量更高的可扩展目标，添加或删除容量将缩小目标值与实际指标数据点之间的差距。
- 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。

多个扩缩策略

- 一个可扩展目标可以具有多个目标跟踪扩展策略，前提是它们分别使用不同的指标。Application 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横

向扩展或横向缩减。如果任何目标跟踪策略已准备好进行向外扩展，它将向外扩展可扩展目标，但仅在所有目标跟踪策略（启用了缩减部分）准备好缩减时才执行缩减。

- 如果多个扩缩策略指示可扩展目标同时横向扩展或横向缩减，则 Application Auto Scaling 会根据为横向缩减和横向扩展提供最大容量的策略进行扩展。这让您能够更灵活地覆盖多种场景，并确保始终有足够的容量来处理工作负载。
- 您可以禁用目标跟踪扩展策略的缩减部分。利用此功能，您可以灵活地使用与用于向外扩展的方法不同的缩减方法。例如，您可以使用步进扩展策略进行缩减，同时使用目标跟踪扩展策略进行横向扩展。
- 不过，在将目标跟踪扩展策略与步进扩展策略结合使用时，我们建议您务必谨慎，因为这些策略之间的冲突可能会导致意外的行为。例如，如果步进扩展策略在目标跟踪策略准备执行缩减之前启动缩减活动，则不会阻止缩减活动。在缩减活动完成后，目标跟踪策略可能会指示可扩展目标重新横向扩展。

在高利用率期间支持应用程序可用性

目标跟踪扩展策略在利用率提高时添加容量比在利用率降低时删除容量更为积极。例如，如果策略的指定指标达到其目标值，则策略假定您的应用程序已达到高负载。因此，它通过尽可能快地添加与指标值成比例的容量来进行响应。指标越高，添加的容量就越多。

当指标低于目标值时，策略预计利用率最终会再次增加。在此场景中，只有当利用率超过远低于目标值（通常比目标值低 10% 以上）的阈值时，它才会通过删除容量来减慢扩缩速度，从而认为利用率已放缓。这种更保守的行为旨在确保只有当应用程序不再遇到与之前相同的高级别需求时，才会删除容量。

对于具有周期性质的工作负载，您还可以选择使用计划扩展按计划自动更改容量。对于每个计划的操作，可以定义新的最小容量值和新的最大容量值。这些值构成扩展策略的边界。

当立即需要容量时，计划扩展和目标跟踪扩展的组合有助于减少利用率级别急剧增加的影响。

选择指标

创建目标跟踪扩缩策略时，可使用以下预定义指标。您可以选择使用自定义指标规范来定义要监控的指标，并将其与目标跟踪扩缩策略一起使用。

AppStream 2.0

- `AppStreamAverageCapacityUtilization`

Aurora

- `RDSReaderAverageCPUUtilization`
- `RDSReaderAverageDatabaseConnections`

Amazon Comprehend

- `ComprehendInferenceUtilization`

DynamoDB

- `DynamoDBReadCapacityUtilization`
- `DynamoDBWriteCapacityUtilization`

Amazon ECS

- `ALBRequestCountPerTarget`（负载均衡器指标）
- `ECSServiceAverageCPUUtilization`
- `ECSServiceAverageMemoryUtilization`

ElastiCache

- `ElastiCachePrimaryEngineCPUUtilization`
- `ElastiCacheReplicaEngineCPUUtilization`
- `ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage`

Amazon Keyspaces (for Apache Cassandra)

- `CassandraReadCapacityUtilization`
- `CassandraWriteCapacityUtilization`

Lambda

- `LambdaProvisionedConcurrencyUtilization`

Amazon Managed Streaming for Apache Kafka (MSK)

- `KafkaBrokerStorageUtilization`

Neptune

- `NeptuneReaderAverageCPUUtilization`

Spot 实例集（Amazon EC2）

- `ALBRequestCountPerTarget`（负载均衡器指标）
- `EC2SpotFleetRequestAverageCPUUtilization`
- `EC2SpotFleetRequestAverageNetworkIn`
- `EC2SpotFleetRequestAverageNetworkOut`

SageMaker

- `SageMakerVariantInvocationsPerInstance`

每个指标表示 Amazon CloudWatch 中存储的一组按时间顺序排列的数据点。尽管 Amazon 中的大多数指标在默认情况下每分钟报告一次，但 Amazon EC2 指标在默认情况下每五分钟报告一次。您可以启用详细监控，从而以一分钟的粒度获取实例的指标数据，但这会产生额外的费用。为了确保在竞价型实例集扩缩时更快地响应利用率变化，我们建议您启用详细监控。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的对实例启用或禁用详细监控。

选择指标时请记住原则：

- 并非所有 指标都适用于目标跟踪。当指定自定义指标时，这可能很重要。指标必须是有效的使用率指标并且描述可扩展目标的繁忙程度。指标值必须根据可扩展目标的容量按比例增加或减少，以便指标数据可用于按比例扩展可扩展目标。
- 要使用 `ALBRequestCountPerTarget` 指标，您必须指定 `ResourceLabel` 参数以标识与该指标关联的目标组。

- 指标向 CloudWatch 发出实际 0 值时（例如，`ALBRequestCountPerTarget`），Application Auto Scaling 在应用程序没有流量时可以横向缩减为 0。要在没有请求路由时将可扩展目标横向缩减到 0，可扩展目标的最小容量必须设置为 0。
- 并非所有服务都可让您通过目标服务的控制台管理自定义指标。要查看目标服务是否支持控制台中的自定义指标，请参阅该服务的文档。
- 当扩缩策略基于吞吐量（例如，每目标的应用程序负载均衡器请求计数、网络 I/O 或其他计数指标）时，目标值表示单个实体（例如 Application Load Balancer 目标组的单个目标）在一分钟内的最佳平均吞吐量。

定义冷却时间

等待上一个扩展活动生效的时间量称为冷却时间。

对于目标跟踪扩展策略，有两种类型的冷却时间：

- 使用 scale-out cooldown period (向外扩展冷却时间)，目的是持续（但不过度）向外扩展。Application Auto Scaling 使用目标跟踪扩展策略成功向外扩展后，它将开始计算冷却时间。除非触发更大的向外扩展或冷却时间结束，否则扩展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。
- 使用 scale-in cooldown period（横向缩减冷却时间），目的是以保守方式进行横向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横向缩减活动。但是，如果另一个警报在缩减冷却时间内触发了向外扩展活动，Application Auto Scaling 将立即向外扩展目标。在这种情况下，缩减冷却时间停止而不完成。

每个冷却时间以秒为单位进行度量，仅适用于与扩展策略相关的扩展活动。在冷却时间内，当计划的操作在计划的时间开始时，它可以立即触发扩展活动，而无需等待冷却时间到期。

您可以从默认值开始，稍后可对其进行微调。例如，您可能需要延长冷却时间，以防止目标跟踪扩展策略对短时间内发生的更改过于激进。有关默认值，请参阅 Application Auto Scaling API 参考中的 TargetTrackingScalingPolicyConfiguration。

扩缩策略创建、管理和删除的常用命令

使用扩缩策略的常用命令包括：

- register-scalable-target 将 Amazon 或自定义资源注册为可扩展目标（Application Auto Scaling 可以扩展的资源），并暂停和恢复扩缩。
- put-scaling-policy 可添加或修改现有可扩展目标的扩缩策略。
- describe-scaling-activities 可返回关于 Amazon 区域中扩缩活动的信息。
- describe-scaling-policies 可返回关于 Amazon 区域中扩缩策略的信息。
- delete-scaling-policy 可删除扩缩策略。

限制

以下是使用目标跟踪扩缩策略时的限制：

- 可扩展目标不能是 Amazon EMR 集群。Amazon EMR 不支持目标跟踪扩缩策略。
- 当 Amazon MSK 集群是可扩展目标时，横向缩减将禁用且无法启用。
- 您不能使用 `RegisterScalableTarget` 或 `PutScalingPolicy` API 操作来更新 Amazon Auto Scaling 扩展计划。有关如何使用扩缩计划的更多信息，请参阅 Amazon Auto Scaling 文档。

使用 Amazon CLI 创建目标跟踪扩缩策略

您可以使用 Amazon CLI 为以下配置任务创建 Application Auto Scaling 的目标跟踪扩缩策略。

任务

注册可扩展目标

如果您尚未注册，请注册可扩展目标。使用 register-scalable-target 命令可将目标服务中的特定资源注册为可扩展目标。以下示例使用 Application Auto Scaling 注册 Spot 实例集请求。Application Auto Scaling 可以扩展 Spot 实例集中的实例数，最少为 2 个实例，最多为 10 个实例。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --min-capacity 2 --max-capacity 10
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace ec2 --scalable-
dimension ec2:spot-fleet-request:TargetCapacity --resource-id spot-fleet-request/
sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --min-capacity 2 --max-capacity 10
```

> **Note**
>
> 为简洁起见，本主题中的示例说明了用于 Amazon EC2 Spot 实例集的 CLI 命令。要指定不同的可扩展目标，请在 `--service-namespace` 中指定其命名空间，在 `--scalable-dimension` 中指定其可扩展维度，并在 `--resource-id` 中指定其资源 ID。

创建目标跟踪扩缩策略

示例：目标跟踪配置文件

以下是将 CPU 平均使用率保持在 40% 的示例目标跟踪配置。将此配置保存在名为 `config.json` 的文件中。

```
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
    }
}
```

有关更多信息，请参阅 Application Auto Scaling API 参考中的 PredefinedMetricSpecification。

或者，您可以通过创建自定义指标规范并为 CloudWatch 的每个参数添加值来使用自定义指标进行扩缩。以下是一个示例目标跟踪配置，它将指定指标的平均利用率保持在 40%。

```
{
```

```
   "TargetValue":40.0,
   "CustomizedMetricSpecification":{
      "MetricName":"MyUtilizationMetric",
      "Namespace":"MyNamespace",
      "Dimensions":[
         {
            "Name":"MyOptionalMetricDimensionName",
            "Value":"MyOptionalMetricDimensionValue"
         }
      ],
      "Statistic":"Average",
      "Unit":"Percent"
   }
}
```

有关更多信息，请参阅 Application Auto Scaling API 参考中的 CustomizedMetricSpecification。

示例：cpu40-target-tracking-scaling-policy

使用以下 put-scaling-policy 命令以及您创建的 `config.json` 文件，创建一个名为 `cpu40-target-tracking-scaling-policy` 的扩缩策略。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scaling-policy --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --policy-name cpu40-target-tracking-scaling-policy --policy-type TargetTrackingScaling \
  --target-tracking-scaling-policy-configuration file://config.json
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace ec2 --scalable-
dimension ec2:spot-fleet-request:TargetCapacity --resource-id spot-fleet-request/
sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --policy-name cpu40-target-tracking-scaling-policy
 --policy-type TargetTrackingScaling --target-tracking-scaling-policy-configuration file://
config.json
```

如果成功，此命令将返回包含代表您创建的两个 CloudWatch 警报的 ARN 和名称。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:policy-id:resource/
ec2/spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE:policyName/cpu40-target-
tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-spot-
fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-
a60f-3b36d653feca",
            "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-
a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
        }
    ]
}
```

描述目标跟踪扩缩策略

您可使用以下 describe-scaling-policies 命令描述指定服务命名空间的所有扩缩策略。

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2
```

您可使用 --query 参数筛选结果以仅显示目标跟踪扩展策略。有关 query 的语法的更多信息，请参阅 Amazon Command Line Interface 用户指南中的控制 Amazon CLI 的命令输出。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2 \
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

Windows

```
aws application-autoscaling describe-scaling-policies --service-namespace ec2 --query
 "ScalingPolicies[?PolicyType==`TargetTrackingScaling`]"
```

下面是示例输出。

```
[
    {
        "PolicyARN": "PolicyARN",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
            },
            "TargetValue": 40.0
        },
        "PolicyName": "cpu40-target-tracking-scaling-policy",
        "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
        "ServiceNamespace": "ec2",
        "PolicyType": "TargetTrackingScaling",
        "ResourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
        "Alarms": [
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-
a60f-3b36d653feca",
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-
spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-
a812-6c67aaf2910d",
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
            }
        ],
        "CreationTime": 1515021724.807
    }
]
```

删除目标跟踪扩缩策略

在使用完目标跟踪扩缩策略后，您可以使用 delete-scaling-policy 命令删除该策略。

以下命令将删除指定 Spot 队组请求的目标跟踪扩展策略。它还将删除 Application Auto Scaling 代表您创建的 CloudWatch 警报。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scaling-policy --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --policy-name cpu40-target-tracking-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace ec2 --scalable-
dimension ec2:spot-fleet-request:TargetCapacity --resource-id spot-fleet-request/
sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --policy-name cpu40-target-tracking-scaling-policy
```

Application Auto Scaling 的分步扩缩策略

借助分步扩缩，您可以为触发扩缩流程的 CloudWatch 警报选择扩缩指标和阈值，并定义当阈值违反指定数量的评估期时应如何扩展您的可扩展目标。

如果您的扩展指标是与可扩展目标的容量成比例增加或减少的利用率指标，我们建议您使用目标跟踪扩展策略。有关更多信息，请参阅 Application Auto Scaling 的目标跟踪扩缩策略 (p. 41)。您仍然可以选择使用目标跟踪扩展与步进扩展来实现更高级的扩展策略配置。例如，如果需要，您可以在利用率达到特定级别时配置更积极的响应。

步进扩展策略根据一组扩展调整增加或减少可扩展目标的当前容量，这些调整称为步进调整。这些调整将根据警报违例规模发生变化。Application Auto Scaling 会在收到警报消息时评估所有违例的警报。

分步调整

创建分步扩展策略时，将添加一个或多个分步调整，让您可以根据警报的严重程度进行扩展。每个分步调整指定以下内容：

- 指标值的下限
- 指标值的上限
- 要扩展的数量（基于扩展调整类型）

CloudWatch 根据与您的 CloudWatch 警报关联的指标的统计信息聚合指标数据点。违反警报时，将触发相应的扩缩策略。Application Auto Scaling 将指定的聚合类型应用于 CloudWatch 的最新指标数据点（而不是原始指标数据）。它将此聚合指标值与步进调整定义的上限和下限进行比较，以确定执行哪个步进调整。

您可以指定相对于违例阈值的上限和下限。例如，假设您有一个既具有当前容量又具有所需容量 10 的可扩展目标。您有一个违规阈值为 50% 的 CloudWatch 警报。您具有调整类型 `PercentChangeInCapacity` 以及以下步进调整的扩展和缩减策略：

示例：扩展策略的步进调整

下限	上限	调整
0	10	0
10	20	10
20	null	30

示例：缩放策略的步进调整

下限	上限	调整
-10	0	0
-20	-10	-10

下限	上限	调整
null	-20	-30

这将创建以下扩展配置。

```
Metric value

-infinity           30%     40%           60%      70%              infinity
-------------------------------------------------------------------------
          -30%       | -10% | Unchanged   | +10%  |        +30%
-------------------------------------------------------------------------
```

以下几点总结了扩展配置相对于可扩展目标的所需容量和当前容量的行为：

- 在聚合指标值大于 40 并小于 60 时，将保留当前容量和所需容量。
- 如果指标值达到 60，Application Auto Scaling 将可扩展目标的所需容量增加 1，达到 11。这基于向外扩展策略的第二个步进调整（增加 10 的 10%）。在增加新容量后，Application Auto Scaling 将当前容量增加到 11。如果即使在增加该容量后指标值仍增加到 70，Application Auto Scaling 会将目标容量增加 3 以达到 14。这基于向外扩展策略的第三个步进调整（增加 11 的 30%，即 3.3，向下舍入到 3）。
- 如果指标值达到 40，根据横向缩减策略的第二个步骤调整（减去 14 的 10%，即 1.4，向下舍入到 1），Application Auto Scaling 将目标容量减少 1 以达到 13。如果在减少该容量后指标值仍降到 30，根据横向缩减策略的第三个步骤调整（减去 13 的 30%，即 3.9，向下舍入到 3），Application Auto Scaling 将目标容量减小 3 以达到 10。

为扩展策略指定步进调整时，请注意以下事项：

- 分步调整范围不能重叠或有间隙。
- 只有一个分步调整可以有空下限 (负无穷)。如果一个分步调整有负下限，则必须有一个分步调整有空下限。
- 只有一个分步调整可以有空上限 (正无穷)。如果一个步进调整有正上限，则必须有一个步进调整有空上限。
- 同一步进调整中的上限和下限不能为空。
- 如果指标值高于违例阈值，则含下限而不含上限。如果指标值低于违例阈值，则不含下限而含上限。

扩展调整类型

您可以根据您选择的扩展调整类型来定义执行最佳扩展操作的扩展策略。您可以将调整类型指定为可扩展目标的当前容量的百分比或绝对数。

对于分步扩缩策略，Application Auto Scaling 支持以下调整类型：

- ChangeInCapacity - 将可扩展目标的当前容量增加或减少指定的值。正值将增加容量，负值将减少容量。例如：如果当前容量为 3 且调整值为 5，则 Application Auto Scaling 将为容量增加 5（总量为 8）。
- ExactCapacity - 将可扩展目标的当前容量更改为指定的值。为该调整类型指定一个正值。例如：如果当前容量为 3 且调整值为 5，则 Application Auto Scaling 将容量更改为 5。
- PercentChangeInCapacity - 将可扩展目标的当前容量增加或减少指定的百分比。正值将增加容量，负值将减少容量。例如：如果当前容量为 10 且调整值为 10%，则 Application Auto Scaling 将为容量增加 1（总量为 11）。

  Note

  如果得出的值不是整数，Application Auto Scaling 将进行舍入，如下所示：

- 大于 1 的值向下取整。例如，12.7 取整为 12。
- 0 和 1 之间的值舍入到 1。例如，.67 取整为 1。
- 0 和 -1 之间的值舍入到 -1。例如，-.58 取整为 -1。
- 小于 -1 的值向上取整。例如，-6.67 取整为 -6。

对于 PercentChangeInCapacity，您还可以使用 MinAdjustmentMagnitude 参数指定最小扩展量。例如，假定您创建一个增加 25% 的策略，并指定最小扩展量为 2。如果可扩展目标的容量为 4 并执行该扩展策略，4 的 25% 为 1。不过，由于您指定最小扩展量为 2，Application Auto Scaling 将增加 2。

冷却时间

等待上一个扩展活动生效的时间量称为冷却时间。

- 使用向外扩展政策，目的是持续（但不过度）向外扩展。Application 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。除非触发更大的向外扩展或冷却时间结束，否则扩展策略不会再次增加所需容量。尽管冷却时间有效，但启动向外扩展活动所添加的容量将计算为下一向外扩展活动所需容量的一部分。例如，如果警报触发一个步进扩展策略以将容量增加 2，将成功完成扩展活动并开始计算冷却时间。如果警报在冷却时间内再次触，但进行了 3 这样更大幅度的步进调整，以前增加的 2 将被视为当前容量的一部分。因此，仅在容量中增加 1。
- 使用横向缩减策略，目的是以保守方式进行横向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横向缩减活动。但是，如果另一个警报在缩减活动后的冷却时间内触发了向外扩展活动，Application Auto Scaling 将立即向外扩展目标。在这种情况下，缩减活动的冷却时间将停止而不完成。

冷却时间以秒为单位进行度量，仅适用于与扩展策略相关的扩展活动。在冷却时间内，当计划的操作在计划的时间开始时，它可以立即触发扩展活动，而无需等待冷却时间到期。

扩缩策略创建、管理和删除的常用命令

使用扩缩策略的常用命令包括：

- register-scalable-target 将 Amazon 或自定义资源注册为可扩展目标（Application Auto Scaling 可以扩展的资源），并暂停和恢复扩缩。
- put-scaling-policy 可添加或修改现有可扩展目标的扩缩策略。
- describe-scaling-activities 可返回关于 Amazon 区域中扩缩活动的信息。
- describe-scaling-policies 可返回关于 Amazon 区域中扩缩策略的信息。
- delete-scaling-policy 可删除扩缩策略。

限制

以下是使用分步扩缩策略时的限制：

- 您无法为某些服务创建分步扩缩策略。DynamoDB、Amazon Comprehend、Lambda、Amazon Keyspaces、Amazon MSK、ElastiCache 或 Neptune 不支持分步扩缩策略。

使用 Amazon CLI 创建分步扩缩策略

您可以使用 Amazon CLI 为以下配置任务创建 Application Auto Scaling 的分步扩缩策略。

任务

注册可扩展目标

如果您尚未注册，请注册可扩展目标。使用 register-scalable-target 命令可将目标服务中的特定资源注册为可扩展目标。以下示例使用 Application Auto Scaling 注册 Amazon ECS 服务。Application Auto Scaling 可扩展任务的数量，最少 2 个任务，最多 10 个任务。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/sample-app-service \
  --min-capacity 2 --max-capacity 10
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace ecs --scalable-
dimension ecs:service:DesiredCount --resource-id service/default/sample-app-service --min-
capacity 2 --max-capacity 10
```

> **Note**
>
> 为简洁起见，本主题中的示例说明了 Amazon ECS 服务的 CLI 命令。要指定不同的可扩展目标，请在 `--service-namespace` 中指定其命名空间，在 `--scalable-dimension` 中指定其可扩展维度，并在 `--resource-id` 中指定其资源 ID。

创建分步扩缩策略

下面是调整类型为 `ChangeInCapacity` 的示例分步配置，其基于以下分步调整增加可扩展目标的容量（假设 CloudWatch 警报阈值为 70%）：

- 当指标的值大于或等于 70% 但小于 85% 时，将容量增加 1
- 当指标的值大于或等于 85% 但小于 95% 时，将容量增加 2
- 当指标值大于或等于 95% 时，将容量增加 3

将此配置保存在名为 `config.json` 的文件中。

```
{
  "AdjustmentType": "ChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "StepAdjustments": [
    {
      "MetricIntervalLowerBound": 0,
      "MetricIntervalUpperBound": 15,
      "ScalingAdjustment": 1
    },
    {
      "MetricIntervalLowerBound": 15,
      "MetricIntervalUpperBound": 25,
```

```
      "ScalingAdjustment": 2
    },
    {
      "MetricIntervalLowerBound": 25,
      "ScalingAdjustment": 3
    }
  ]
}
```

使用以下 put-scaling-policy 命令以及您创建的 config.json 文件创建一个名为 my-step-scaling-policy 的扩缩策略。

Linux、macOS 或 Unix

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/sample-app-service \
  --policy-name my-step-scaling-policy --policy-type StepScaling \
  --step-scaling-policy-configuration file://config.json
```

Windows

```
aws application-autoscaling put-scaling-policy --service-namespace ecs --scalable-
dimension ecs:service:DesiredCount --resource-id service/default/sample-app-service
 --policy-name my-step-scaling-policy --policy-type StepScaling --step-scaling-policy-
configuration file://config.json
```

输出包括作为策略唯一名称的 ARN。您需要使用它来创建 CloudWatch 警报。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-
cbeb-4294-891c-a5a941dfa787:resource/ecs/service/default/sample-app-service:policyName/my-
step-scaling-policy"
}
```

创建触发扩缩策略的警报

最后，使用以下 CloudWatch put-metric-alarm 命令创建要与分步扩缩策略一起使用的警报。在本示例中，您将根据平均 CPU 利用率发出警报。如果警报在至少两个连续 60 秒的评估期间达到 70% 的阈值，则它将被配置为处于 ALARM 状态。要指定其他 CloudWatch 指标或使用您自己的自定义指标，请在 --metric-name 中指定其名称并在 --namespace 中指定其命名空间。

Linux、macOS 或 Unix

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-ECS:service/default/
sample-app-service \
  --metric-name CPUUtilization --namespace AWS/ECS --statistic Average \
  --period 60 --evaluation-periods 2 --threshold 70 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions Name=ClusterName,Value=default Name=ServiceName,Value=sample-app-service \
  --alarm-actions PolicyARN
```

Windows

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-ECS:service/
default/sample-app-service --metric-name CPUUtilization --namespace AWS/ECS --
statistic Average --period 60 --evaluation-periods 2 --threshold 70 --comparison-
```

```
operator GreaterThanOrEqualToThreshold --dimensions Name=ClusterName,Value=default
 Name=ServiceName,Value=sample-app-service --alarm-actions PolicyARN
```

描述分步扩缩策略

您可使用以下 describe-scaling-policies 命令描述指定服务命名空间的所有扩缩策略。

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

您可以使用 `--query` 参数将结果筛选为仅步进扩展策略。有关 `query` 的语法的更多信息，请参阅 Amazon Command Line Interface 用户指南中的控制 Amazon CLI 的命令输出。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs \
  --query 'ScalingPolicies[?PolicyType==`StepScaling`]'
```

Windows

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs --query
 "ScalingPolicies[?PolicyType==`StepScaling`]"
```

下面是示例输出。

```
[
    {
        "PolicyARN": "PolicyARN",
        "StepScalingPolicyConfiguration": {
            "MetricAggregationType": "Average",
            "Cooldown": 60,
            "StepAdjustments": [
                {
                    "MetricIntervalLowerBound": 0.0,
                    "MetricIntervalUpperBound": 15.0,
                    "ScalingAdjustment": 1
                },
                {
                    "MetricIntervalLowerBound": 15.0,
                    "MetricIntervalUpperBound": 25.0,
                    "ScalingAdjustment": 2
                },
                {
                    "MetricIntervalLowerBound": 25.0,
                    "ScalingAdjustment": 3
                }
            ],
            "AdjustmentType": "ChangeInCapacity"
        },
        "PolicyType": "StepScaling",
        "ResourceId": "service/default/sample-app-service",
        "ServiceNamespace": "ecs",
        "Alarms": [
            {
                "AlarmName": "Step-Scaling-AlarmHigh-ECS:service/default/sample-app-
service",
                "AlarmARN": "arn:aws:cloudwatch:region:012345678910:alarm:Step-Scaling-
AlarmHigh-ECS:service/default/sample-app-service"
            }
        ],
```

```
        "PolicyName": "my-step-scaling-policy",
        "ScalableDimension": "ecs:service:DesiredCount",
        "CreationTime": 1515024099.901
    }
]
```

删除分步扩缩策略

当您不再需要某个步进扩展策略时，可将其删除。要删除的扩缩策略和 CloudWatch 警报，请完成以下任务。

删除您的扩展策略

使用以下 delete-scaling-policy 命令。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scaling-policy --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/sample-app-service \
  --policy-name my-step-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace ecs --scalable-
dimension ecs:service:DesiredCount --resource-id service/default/sample-app-service --
policy-name my-step-scaling-policy
```

删除 CloudWatch 警报

使用 delete-alarms 命令。您可以一次删除一个或多个警报。例如，使用以下命令可删除 `Step-Scaling-AlarmHigh-ECS:service/default/sample-app-service` 和 `Step-Scaling-AlarmLow-ECS:service/default/sample-app-service` 警报：

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-ECS:service/default/
sample-app-service Step-Scaling-AlarmLow-ECS:service/default/sample-app-service
```

教程：配置弹性伸缩以处理繁重的工作负载

Important

在开始学习本教程之前，我们建议您首先阅读以下介绍性教程：教程：使用 Amazon CLI 的计划扩缩入门 (p. 34)。

在本教程中，您将了解当应用程序的工作负载比正常工作负载重的情况下，如何根据时间范围进行横向扩展和横向缩减。当您的应用程序可能会突然有大量定期或季节性的访问者时，这会很有帮助。

您可以将目标跟踪扩缩策略与计划的扩缩一起使用来处理额外负载。计划的扩缩会根据您指定的计划代表您自动启动对 `MinCapacity` 和 `MaxCapacity` 的更改。当目标跟踪扩缩策略在资源上处于活动状态时，它可以根据当前资源利用率在新的最小容量和最大容量范围内动态扩展。

完成本教程后，您将了解如何：

- 使用计划的扩缩添加额外容量，以在达到限值之前满足重负载，然后在不再需要时删除额外容量。
- 使用目标跟踪扩缩策略根据当前资源利用率扩展您的应用程序。

目录

先决条件

本教程假定您已执行以下操作：

- 您已创建 Amazon 账户。有关更多信息，请参阅注册 Amazon Web Services 账户 (p. 3)。
- 您已经安装并配置 Amazon CLI。有关更多信息，请参阅设置 Amazon CLI (p. 3)。
- 您的账户具有使用 Application Auto Scaling 将资源注册和取消注册为可扩展目标的所有必要权限。它还具有创建扩缩策略和计划的操作的所有必要权限。有关更多信息，请参阅适用于 Application Auto Scaling 的 Identity and Access Management (p. 74)。
- 您在非生产环境中拥有可用于本教程的受支持资源。如果您还没有，请立即创建一个。有关使用 Application Auto Scaling 的 Amazon 服务和资源的信息，请参阅 可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7) 部分。

Note

完成本教程时有两个步骤，您可在这两个步骤中将最大和最小容量值设置为 0，以将当前容量重置为 0。根据您当前通过 Application Auto Scaling 使用的资源，您可能无法在这些步骤中将当前容量

设置为 0。为帮助您解决此问题，输出中将会显示一条消息，指示最小容量不能小于指定的值，并将提供 Amazon 资源可接受的最小容量值。

步骤 1：注册您的可扩展目标

首先使用 Application Auto Scaling 将您的资源注册为可扩展目标。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。

向 Application Auto Scaling 注册您的可扩展目标

- 使用以下 register-scalable-target 命令注册新的可扩展目标。将 `--min-capacity` 和 `--max-capacity` 值设置为 0 以将当前容量重置为 0。

  将 `--service-namespace` 中的示例文本替换为您通过 Application Auto Scaling 使用的 Amazon 服务的命名空间，将 `--scalable-dimension` 替换为与您注册的资源关联的可扩展维度，并将 `--resource-id` 替换为资源的标识符。这些值因使用的资源以及资源 ID 的构造方式而异。有关更多信息，请参阅 可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7) 部分中的主题。这些主题包括向您显示如何使用 Application Auto Scaling 注册可扩展目标的示例命令。

  Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --min-capacity 0 --max-capacity 0
```

  Windows

```
aws application-autoscaling register-scalable-target --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --min-capacity 0 --max-capacity 0
```

  如果此命令成功执行，将不会返回任何输出。

步骤 2：根据您的要求设置计划的操作

您可以使用 put-scheduled-action 命令创建配置为满足业务需求的计划操作。在本教程中，我们重点介绍一种配置，通过将容量减少到 0 来停止在工作时间以外消耗资源。

创建在噪声横向扩展的计划操作

1. 要横向扩展可扩展目标，请使用以下 put-scheduled-action 命令。使用 cron 表达式将 `--schedule` 参数包含在采用 UTC 时间的定期计划中。

   根据指定的计划（UTC 时间每天上午 9:00），Application Auto Scaling 会将 `MinCapacity` 和 `MaxCapacity` 值更新为 1-5 个容量单位的所需范围。

   Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --scheduled-action-name my-first-scheduled-action \
```

```
  --schedule "cron(0 9 * * ? *)" \
  --scalable-target-action MinCapacity=1,MaxCapacity=5
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --scheduled-action-name my-
first-scheduled-action --schedule "cron(0 9 * * ? *)" --scalable-target-action
 MinCapacity=1,MaxCapacity=5
```

如果此命令成功执行，将不会返回任何输出。

2. 要确认您的计划操作是否存在，请使用以下 describe-scheduled-actions 命令。

   Linux、macOS 或 Unix

```
aws application-autoscaling describe-scheduled-actions \
  --service-namespace namespace \
  --query 'ScheduledActions[?ResourceId==`identifier`]'
```

   Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace namespace --
query "ScheduledActions[?ResourceId==`identifier`]"
```

   下面是示例输出。

```
[
    {
        "ScheduledActionName": "my-first-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 9 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 1,
            "MaxCapacity": 5
        },
        ...
    }
]
```

创建在夜间横向缩减的计划操作

1. 重复上述过程以创建另一个计划的操作，Application Auto Scaling 使用该操作在当天结束时进行横向缩减。

   根据指定的计划（UTC 时间每天晚上 8:00），Application Auto Scaling 会将目标的 `MinCapacity` 和 `MaxCapacity` 更新为 0，如以下 put-scheduled-action 命令所示。

   Linux、macOS 或 Unix

```
aws application-autoscaling put-scheduled-action \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --scheduled-action-name my-second-scheduled-action \
  --schedule "cron(0 20 * * ? *)" \
  --scalable-target-action MinCapacity=0,MaxCapacity=0
```

Windows

```
aws application-autoscaling put-scheduled-action --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --scheduled-action-name my-
second-scheduled-action --schedule "cron(0 20 * * ? *)" --scalable-target-action
 MinCapacity=0,MaxCapacity=0
```

2. 要确认您的计划操作是否存在，请使用以下 describe-scheduled-actions 命令。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scheduled-actions \
  --service-namespace namespace \
  --query 'ScheduledActions[?ResourceId==`identifier`]'
```

Windows

```
aws application-autoscaling describe-scheduled-actions --service-namespace namespace --
query "ScheduledActions[?ResourceId==`identifier`]"
```

下面是示例输出。

```
[
    {
        "ScheduledActionName": "my-first-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 9 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 1,
            "MaxCapacity": 5
        },
        ...
    },
    {
        "ScheduledActionName": "my-second-scheduled-action",
        "ScheduledActionARN": "arn",
        "Schedule": "cron(0 20 * * ? *)",
        "ScalableTargetAction": {
            "MinCapacity": 0,
            "MaxCapacity": 0
        },
        ...
    }
]
```

步骤 3：添加目标跟踪扩缩策略

现在，您已经准备好基本计划，请添加一个目标跟踪扩缩策略，以根据当前资源利用率进行扩展。

通过目标跟踪，Application Auto Scaling 会将策略中的目标值与指定指标的当前值进行比较。如果两个值在一段时间内不相等，Application Auto Scaling 会添加或删除容量以保持稳定的性能。随着应用程序负载和指标值的增加，Application Auto Scaling 会尽可能快地增加容量，而不会超过 `MaxCapacity`。当 Application Auto Scaling 由于负载最小而删除容量时，它会在不低于 `MinCapacity` 时执行此操作。通过根据使用情况调整容量，您只需支付应用程序所需的费用。有关更多信息，请参阅在高利用率期间支持应用程序可用性 (p. 42)。

如果由于您的应用程序没有任何负载而导致指标数据不足，则 Application Auto Scaling 不会添加或删除容量。此行为的目的是在信息不足的情况下优先考虑可用性。

您可以添加多个扩缩策略，但请确保不会添加冲突的分步扩缩策略，这可能会导致不良行为。例如，如果步进扩展策略在目标跟踪策略准备执行缩减之前启动缩减活动，则不会阻止缩减活动。在横向缩减活动完成后，目标跟踪策略可能会指示 Application Auto Scaling 再次横向扩展。

创建目标跟踪扩展策略

1. 使用以下 put-scaling-policy 命令创建策略。

   最常用于目标跟踪的指标是预定义的，您可以在不提供 CloudWatch 的完整指标规范的情况下使用这些指标。有关可用预定义指标的更多信息，请参阅 Application Auto Scaling 的目标跟踪扩缩策略 (p. 41)。

   在运行此命令之前，请确保您的预定义指标需要目标值。例如，要在 CPU 利用率达到 50% 时横向扩展，请将目标值指定为 50.0。或者，要在使用率达到 70% 时横向扩展 Lambda 预置并发，请将目标值指定为 0.7。有关特定资源目标值的信息，请参阅服务提供的有关如何配置目标跟踪的文档。有关更多信息，请参阅可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7)。

   Linux、macOS 或 Unix

   ```
   aws application-autoscaling put-scaling-policy \
     --service-namespace namespace \
     --scalable-dimension dimension \
     --resource-id identifier \
     --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
     --target-tracking-scaling-policy-configuration '{ "TargetValue": 50.0,
    "PredefinedMetricSpecification": { "PredefinedMetricType": "predefinedmetric" }}'
   ```

   Windows

   ```
   aws application-autoscaling put-scaling-policy --service-namespace namespace --
   scalable-dimension dimension --resource-id identifier --policy-name my-scaling-policy
    --policy-type TargetTrackingScaling --target-tracking-scaling-policy-configuration
    "{ \"TargetValue\": 50.0, \"PredefinedMetricSpecification\": { \"PredefinedMetricType
   \": \"predefinedmetric\" }}"
   ```

   如果成功，此命令将返回包含代表您创建的两个 CloudWatch 警报的 ARN 和名称。

2. 要确认您的计划操作是否存在，请使用以下 describe-scaling-policies 命令。

   Linux、macOS 或 Unix

   ```
   aws application-autoscaling describe-scaling-policies --service-namespace namespace \
     --query 'ScalingPolicies[?ResourceId==`identifier`]'
   ```

   Windows

   ```
   aws application-autoscaling describe-scaling-policies --service-namespace namespace --
   query "ScalingPolicies[?ResourceId==`identifier`]"
   ```

   下面是示例输出。

   ```
   [
       {
           "PolicyARN": "arn",
           "TargetTrackingScalingPolicyConfiguration": {
               "PredefinedMetricSpecification": {
                   "PredefinedMetricType": "predefinedmetric"
   ```

```
            },
            "TargetValue": 50.0
        },
        "PolicyName": "my-scaling-policy",
        "PolicyType": "TargetTrackingScaling",
        "Alarms": [],
        ...
    }
]
```

步骤 4：后续步骤

发生扩缩活动时，您可以在可扩展目标的扩缩活动输出中看到该活动的记录，例如：

```
Successfully set desired count to 1. Change successfully fulfilled by ecs.
```

要使用 Application Auto Scaling 监控扩缩活动，您可以使用以下 describe-scaling-activities 命令。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-activities
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace namespace --
scalable-dimension dimension --resource-id identifier
```

第 5 步：清除

为防止您的账户对主动扩缩时创建的资源产生费用，您可以按如下方式清理关联的扩缩配置。

删除扩缩配置不会删除您基础的 Amazon 资源。该操作也不会恢复为其原来的容量。您可以使用创建资源的服务的控制台来删除该资源或调整其容量。

删除计划的操作

以下 delete-scheduled-action 命令可删除指定的计划操作。如果您要保留创建的计划操作，您可以跳过此步骤。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scheduled-action \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --scheduled-action-name my-second-scheduled-action
```

Windows

```
aws application-autoscaling delete-scheduled-action --service-namespace namespace --
scalable-dimension dimension --resource-id identifier --scheduled-action-name my-second-
scheduled-action
```

删除扩缩策略

以下 delete-scaling-policy 命令可删除指定的目标跟踪扩缩策略。如果您要保留创建的扩缩策略，您可以跳过此步骤。

Linux、macOS 或 Unix

```
aws application-autoscaling delete-scaling-policy \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --policy-name my-scaling-policy
```

Windows

```
aws application-autoscaling delete-scaling-policy --service-namespace namespace --scalable-
dimension dimension --resource-id identifier --policy-name my-scaling-policy
```

撤消可扩展目标的注册

使用以下 deregister-scalable-target 命令可取消注册可扩展目标。如果您有任何您创建的扩展策略或尚未删除的计划操作，这条命令会将它们删除。如果您要将此可扩展目标保留供将来使用，您可以跳过此操作。

Linux、macOS 或 Unix

```
aws application-autoscaling deregister-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier
```

Windows

```
aws application-autoscaling deregister-scalable-target --service-namespace namespace --
scalable-dimension dimension --resource-id identifier
```

暂停和恢复 Application Auto Scaling 的扩缩

本主题说明如何暂停然后恢复应用程序中可扩展目标的一个或多个扩展活动。暂停-恢复功能用于临时暂停由您的扩展策略和计划操作触发的扩展活动。暂停-恢复功能非常有用，例如，当您更改或调查配置问题时，不希望自动扩展潜在产生干扰。您可以保留您的扩展策略和计划操作，在您准备就绪时，可以恢复扩展活动。

在随后的示例 CLI 命令中，您可在 config.json 文件中传递 JSON 格式的参数。您还可以通过使用引号将 JSON 数据结构括起来，在命令行上传递这些参数。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的在 Amazon CLI 中将引号和字符串结合使用。

目录

扩缩活动

Application Auto Scaling 支持将以下扩缩活动置于暂停状态：

- 由扩展策略触发的所有缩减活动。
- 由扩展策略触发的所有横向扩展活动。
- 涉及计划操作的所有扩展活动。

以下描述说明了暂停各个扩展活动时会发生什么。每个扩展活动都可以单独暂停和恢复。根据暂停扩展活动的原因，您可能需要一起暂停多个扩展活动。

`DynamicScalingInSuspended`

- 在触发目标跟踪扩缩策略或分步扩缩策略时，Application Auto Scaling 不会删除容量。这使您可以暂时禁用与扩缩策略关联的横向缩减活动，而不删除扩缩策略或其关联的 CloudWatch 警报。当您恢复横向缩减时，Application Auto Scaling 会评估具有当前违反的警报阈值的策略。

`DynamicScalingOutSuspended`

- 在触发目标跟踪扩缩策略或分步扩缩策略时，Application Auto Scaling 不会增加容量。这使您可以暂时禁用与扩缩策略关联的横向扩展活动，而不删除扩缩策略或其关联的 CloudWatch 警报。当您恢复横向扩展时，Application Auto Scaling 会评估具有当前违反的警报阈值的策略。

`ScheduledScalingSuspended`

- 在暂停期间，Application Auto Scaling 不启动计划要运行的扩缩操作。当您恢复计划的扩缩时，Application Auto Scaling 仅评估尚未经过执行时间的计划操作。

使用 Amazon CLI 暂停和恢复扩缩活动

您可以暂停和恢复 Application Auto Scaling 可扩展目标的单个或所有扩缩活动。

> **Note**
>
> 为简洁起见，这些示例说明了如何暂停和恢复 DynamoDB 表的扩缩。要指定不同的可扩展目标，请在 `--service-namespace` 中指定其命名空间，在 `--scalable-dimension` 中指定其可扩展维度，并在 `--resource-id` 中指定其资源 ID。

暂停扩展活动

打开一个命令行窗口，然后使用 register-scalable-target 命令和 `--suspended-state` 选项，如下所示。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

要仅暂停某个扩展策略触发的缩减活动，请在 config.json 中指定以下内容。

```
{
    "DynamicScalingInSuspended":true
}
```

要仅暂停某个扩展策略触发的横向扩展活动，请在 config.json 中指定以下内容。

```
{
    "DynamicScalingOutSuspended":true
}
```

要仅暂停涉及计划操作的扩展活动，请在 config.json 中指定以下内容。

```
{
    "ScheduledScalingSuspended":true
}
```

暂停所有扩展活动

使用 register-scalable-target 命令和 `--suspended-state` 选项，如下所示。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

此示例假定文件 config.json 包含以下 JSON 格式的参数。

```
{
    "DynamicScalingInSuspended":true,
    "DynamicScalingOutSuspended":true,
    "ScheduledScalingSuspended":true
}
```

查看暂停的扩缩活动

使用 describe-scalable-targets 命令可确定可扩展目标处于暂停状态的扩缩活动。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scalable-targets --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table
```

Windows

```
aws application-autoscaling describe-scalable-targets --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table
```

下面是示例输出。

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "dynamodb",
            "ScalableDimension": "dynamodb:table:ReadCapacityUnits",
            "ResourceId": "table/my-table",
            "MinCapacity": 1,
            "MaxCapacity": 20,
            "SuspendedState": {
                "DynamicScalingOutSuspended": true,
                "DynamicScalingInSuspended": true,
                "ScheduledScalingSuspended": true
            },
            "CreationTime": 1558125758.957,
            "RoleARN": "arn:aws:iam::123456789012:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable"
        }
    ]
}
```

恢复扩缩活动

当您准备好恢复扩缩活动时，可以使用 register-scalable-target 命令恢复它。

以下示例命令恢复指定的可扩展目标的所有扩展活动。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table \
  --suspended-state file://config.json
```

Windows

```
aws application-autoscaling register-scalable-target --service-namespace dynamodb --
scalable-dimension dynamodb:table:ReadCapacityUnits --resource-id table/my-table --
suspended-state file://config.json
```

此示例假定文件 config.json 包含以下 JSON 格式的参数。

```
{
    "DynamicScalingInSuspended":false,
    "DynamicScalingOutSuspended":false,
    "ScheduledScalingSuspended":false
}
```

Application Auto Scaling 监控

监控是保持 Application Auto Scaling 和其他 Amazon 解决方案的可靠性、可用性和性能的重要方面。您应该从 Amazon 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）。Amazon 提供监控工具以监控 Application Auto Scaling，当出现问题时报告并在适当时采取自动措施。

您可以使用以下功能来帮助您管理 Amazon 资源：

Amazon CloudWatch 警报

CloudWatch 可以自动监控 Amazon 资源的特定指标，从而帮助您检测不正常的应用程序行为。您可以配置 CloudWatch 警报并设置 Amazon SNS 通知，以在指标值不符合预期或者检测到特定异常时发送电子邮件。例如，您可以在网络活动突然高于或低于指标的预期值时收到通知。有关更多信息，请参阅 使用 CloudWatch 警报进行监控 (p. 67) 和 Amazon CloudWatch 用户指南。

Amazon CloudWatch 控制面板

Amazon CloudWatch 实时监控您的 Amazon 资源以及在 Amazon 上运行的应用程序。CloudWatch 控制面板是 CloudWatch 控制台中可自定义的主页。您可以使用这些页面在单个视图中监控您的资源，甚至包括分布在不同区域的资源。您可以使用 CloudWatch 控制面板创建 Amazon 资源的指标和警报的自定义视图。有关更多信息，请参阅使用 CloudWatch 构建控制面板 (p. 68)。

Amazon CloudWatch Events

使用 CloudWatch Events，您可以编写规则，用于根据 Application Auto Scaling 对 API 的调用结果在其他 Amazon 中触发自动操作。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南中的使用 Amazon CloudTrail创建触发 Amazon API 调用的 CloudWatch Events 规则。

Amazon CloudTrail

Amazon CloudTrail 捕获由某个 Amazon Web Services 账户 发出或代表该账户发出的 API 调用和相关事件。然后它将日志文件传送到您指定的 Amazon S3 存储桶。您可以标识哪些用户和账户调用了 Amazon、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 Amazon CloudTrail 用户指南。有关由 CloudTrail 记录的 Application Auto Scaling API 调用的信息，请参阅使用 CloudTrail 记录 Application Auto Scaling API 调用。

Amazon CloudWatch Logs

Amazon CloudWatch Logs 使您能够监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。

使用 CloudWatch 警报进行监控

您可以创建警报，以在 Amazon CloudWatch 检测到可能需要您注意的任何问题时通知您。CloudWatch 可帮助您自动监控 Amazon 的特定指标。

CloudWatch 警报会监控一个指标。该警报仅当状态发生变化并且已持续您指定的时间段时才会触发一个或多个操作。例如，您可以设置一个警报以在指标值低于或超过特定水平时通知您，从而确保在潜在问题出现之前您就得到通知。

CloudWatch 还允许您设置警报，当指标处于 `INSUFFICIENT_DATA` 状态时通知您。任何 Amazon 服务的任何指标均可对 `INSUFFICIENT_DATA` 发出警报。这是新警报的初始状态，但如果 CloudWatch 指标变为不可用，或没有足够的数据可用于指标以确定警报状态时，警报状态也会变为 `INSUFFICIENT_DATA`。例如，仅当 Lambda 函数处于活动状态时，Amazon Lambda 才会每分钟向 CloudWatch 发出一次 `ProvisionedConcurrencyUtilization` 指标。如果函数处于非活动状态，则会导致警报在等待指标时

进入 INSUFFICIENT_DATA 状态。这是正常的，可能不一定意味着存在问题，但如果您预期在一段时间内进行活动，但没有任何活动，则可能表明存在问题。

本主题介绍如何创建警报，以在指标处于您定义的阈值范围之内或之外时或数据不足时发送通知。有关更多详细信息，请参阅 Amazon CloudWatch 用户指南中的使用 Amazon CloudWatch 警报。

创建发送电子邮件的警报

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，依次选择 Alarms 和 Create Alarm。
3. 选择 Select Metric（选择指标）。

   系统会将您导向到可在其中找到所有指标的页面。可用指标的类型取决于您使用的服务和功能。指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。
4. 选择一个指标命名空间（例如 Lambda），然后选择一个指标维度（例如 By Function Name [按函数名称]）。

   All metrics（所有指标）选项卡显示所选维度和命名空间的所有指标。
5. 选中您要为其创建警报的指标旁边的复选框，然后选择 Select metric（选择指标）。
6. 按如下所示配置警报，然后选择 Next（下一步）：

   - 在 Metric（指标）下，选择 1 minute 或 5 minutes 的汇总期。如果您使用一分钟作为某个指标的汇总期，则每分钟具有一个数据点。周期越短，创建的警报越敏感。
   - 在 Conditions（条件）下，配置您的阈值，例如，生成通知之前指标必须超过的值。
   - 在 Additional configuration（其他配置）下，对于 Datapoints to alarm（触发警报的数据点数），输入指标值必须满足阈值条件才会触发警报的数据点（评估时间段）数。例如，2 个连续的 5 分钟时间段需要花 10 分钟才会触发警报。
   - 对于 Missing data treatment（缺失数据处理），保留默认值并将缺失的数据点处理为缺失。

     某些指标仅在发生活动时报告。这可能会导致报告稀疏的指标。如果指标在设计上经常缺少数据点，则在这些期间警报的状态为 INSUFFICIENT_DATA。要强制警报保持之前的 ALARM 或 OK 状态以防止警报摆动，您可以选择忽略缺少的数据。
7. 在 Notification（通知）下，选择警报处于 ALARM、OK 或 INSUFFICIENT_DATA 状态时通知的 SNS 主题。要使告警为相同告警状态或不同告警状态发送多个通知，请选择 Add notification（添加通知）。
8. 在完成后，选择下一步。
9. 输入警报的名称和描述（可选），然后选择 Next (下一步)。
10. 选择Create alarm（创建警报）。

检查警报的状态

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 Alarms（告警）以查看警报列表。
3. 要筛选警报，请使用搜索字段旁边的下拉筛选器，然后选择要应用的筛选选项。
4. 要编辑或删除警报，请选择警报，然后选择 Actions（操作）、Edit（编辑） 或 Actions（操作）、Delete（删除）。

使用 CloudWatch 构建控制面板

您可以使用 Amazon CloudWatch（它将生成有关您的使用情况和性能的指标）监控应用程序使用资源的方式。CloudWatch 从您的 Amazon 资源和您在 Amazon 上运行的应用程序收集原始数据，并将其处理为可读的近实时指标。这些指标保留 15 个月，以便您可以访问历史信息，从而更好地了解应用程序的执行情况。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

CloudWatch 控制面板是 CloudWatch 控制台中的可自定义主页，可用于在单个视图中监控资源，即便是分布到不同区域的资源，也能对其进行监控。您可以使用 CloudWatch 控制面板创建 Amazon 资源的所选指标的自定义视图。您可以在每个图表上选择用于每个指标的颜色，以便更轻松地跨多个图表跟踪同一指标。

创建 CloudWatch 控制面板

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择控制面板，然后选择创建新的控制面板。
3. 输入控制面板的名称，例如要查看其 CloudWatch 数据的服务的名称。
4. 请选择创建控制面板。
5. 选择要添加到控制面板的小部件类型，例如折线图。然后选择配置，并选择要添加到控制面板的指标。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的在 CloudWatch 控制面板中添加或删除图表。

默认情况下，您在 CloudWatch 控制面板中创建的指标为平均值。

指标与维度

当您和与 Application Auto Scaling 集成的服务进行交互时，它们会将下表中显示的指标发送到 CloudWatch。在 CloudWatch 中，指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

这些指标可帮助您发现应用程序的容量要求。您可以使用此信息静态设置容量或设置自动扩展。如果您的应用程序的工作负载不稳定，则表明您应该考虑使用弹性伸缩。

指标名称	命名空间	维度	适用于	
AvailableCapacity	AWS/AppStream	队列	AppStream	
CapacityUtilization	AWS/AppStream	队列	AppStream	
CPU 利用率	AWS/RDS	DBClusterIdentifier、Role (READER)	Aurora	
DatabaseConnections	AWS/RDS	DBClusterIdentifier、Role (READER)	Aurora	
InferenceUtilization	AWS/Comprehend	EndpointArn	Comprehend	
已配置读取容量单位	AWS/DynamoDB	TableName、GlobalSecondaryIndexName	DynamoDB	
已配置写入容量单位	AWS/DynamoDB	TableName、GlobalSecondaryIndexName	DynamoDB	
已使用读取容量单位	AWS/DynamoDB	TableName、GlobalSecondaryIndexName	DynamoDB	
已使用写入容量单位	AWS/DynamoDB	TableName、GlobalSecondaryIndexName	DynamoDB	
CPU 利用率	AWS/ECS	ClusterName、ServiceName	ECS	

指标名称	命名空间	维度	适用于	
MemoryUtilization	AWS/ECS	ClusterName、ServiceName	ECS	
RequestCountPerTarget	AWS/ApplicationELB	TargetGroup	ECS	
EngineCPUUtilization	AWS/ElastiCache	ReplicationGroupId，角色（主要）	ElastiCache	
EngineCPUUtilization	AWS/ElastiCache	ReplicationGroupId，角色（副本）	ElastiCache	
YARNMemoryAvailablePercentage	AWS/ElasticMapReduce	ClusterId	EMR	
已配置读取容量单位	AWS/Cassandra	Keyspace，TableName	Amazon Keyspaces	
已配置写入容量单位	AWS/Cassandra	Keyspace，TableName	Amazon Keyspaces	
已使用读取容量单位	AWS/Cassandra	Keyspace，TableName	Amazon Keyspaces	
已使用写入容量单位	AWS/Cassandra	Keyspace，TableName	Amazon Keyspaces	
ProvisionedConcurrencyUtilization	AWS/Lambda	FunctionName、Resource	Lambda	
KafkaDataLogsDiskUsed	AWS/Kafka	集群名称	Amazon MSK	
KafkaDataLogsDiskUsed	AWS/Kafka	集群名称，代理 ID	Amazon MSK	
CPU 利用率	AWS/Neptune	DBClusterIdentifier、Role (READER)	Neptune	
InvocationsPerInstance	AWS/SageMaker	EndpointName、VariantName	SageMaker	
CPU 利用率	AWS/EC2Spot	FleetRequestId	竞价型实例集	
NetworkIn	AWS/EC2Spot	FleetRequestId	竞价型实例集	
网络输出	AWS/EC2Spot	FleetRequestId	竞价型实例集	
RequestCountPerTarget	AWS/ApplicationELB	TargetGroup	竞价型实例集	

虽然 CloudWatch 允许您为每个指标选择任何统计数据和周期，但并非所有的组合都有用。例如，CPU 利用率的平均、最小和最大统计数据均有用，但求和统计数据却无用。有关更多信息，请参阅上表提供的链接中的服务文档。

衡量应用程序性能的常用方法是平均 CPU 利用率。如果 CPU 利用率增加，而您没有足够的容量来处理它，则应用程序可能无响应。另一方面，如果在利用率低时您有太多的容量和资源正在运行，这会增加使用该服务的成本。

根据服务的不同，您还拥有跟踪可用预配置吞吐量的指标。例如，对于在具有预置并发性的函数别名或版本上正在处理的调用数，Lambda 会发出 `ProvisionedConcurrencyUtilization` 指标。如果您正在启动大型作业并同时多次调用同一函数，则当作业超出可用的预配置并发性时，该作业可能会遇到延迟。另一方面，如果您的预配置并发性比您需要的多，则您的成本可能高于应有的成本。

如果您在 CloudWatch 控制台中未看到这些指标，请确保您已完成资源的设置。在完全设置资源之前，不会显示指标。此外，如果某个指标在过去 14 天内未发布数据，在搜索要添加到 CloudWatch 控制面板上的图表的指标时，将找不到该指标。有关如何手动添加任何指标的信息，请参阅 Amazon CloudWatch 用户指南中的在 CloudWatch 控制面板上手动绘制指标。

Application Auto Scaling 中的安全性

Amazon 十分重视云安全性。作为 Amazon 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Amazon和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础设施。Amazon 还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，第三方审计人员将定期测试和验证安全措施的有效性。要了解适用于 Application Auto Scaling 的合规性计划，请参阅合规性计划范围内的 Amazon 服务。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，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 Application Auto Scaling 时应用责任共担模式。以下主题说明如何配置 Application Auto Scaling 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 Amazon 服务以帮助您监控和保护 Application Auto Scaling 资源。

主题

Application Auto Scaling 和接口 VPC 终端节点

您可以通过将 Application Auto Scaling 配置为使用接口 VPC 端点，以改善 VPC 的安保状况。接口端点由 Amazon PrivateLink 提供支持，此技术让您能够将 VPC 和 Application Auto Scaling 之间的所有网络流量限制在 Amazon 网络内，从而实现对 Application Auto Scaling API 的私有访问。借助接口终端节点，您也不需要 Internet 网关、NAT 设备或虚拟专用网关。

不要求您配置 Amazon PrivateLink，但推荐进行配置。有关 Amazon PrivateLink 和 VPC 终端节点的更多信息，请参阅《Amazon PrivateLink 指南》中的什么是 Amazon PrivateLink？

主题

创建接口 VPC 终端节点

使用以下服务名称为 Application Auto Scaling 创建端点：

```
com.amazonaws.region.application-autoscaling
```

有关更多信息，请参阅 Amazon PrivateLink 指南中的使用接口 VPC 端点访问 Amazon 服务。

您不需要更改任何其他设置。Application Auto Scaling 使用服务终端节点或私有接口 VPC 终端节点（二者中在使用中的那个）调用其他 Amazon 服务。

创建 VPC 终端节点策略

您可以向 VPC 终端节点附加策略来控制对 Application Auto Scaling API 的访问。该策略指定：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

以下示例显示了一个 VPC 终端节点策略，该策略拒绝所有人通过终端节点删除扩展策略的权限。示例策略还授予所有人执行所有其他操作的权限。

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "application-autoscaling:DeleteScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

有关更多信息，请参阅《Amazon PrivateLink 指南》中的 VPC 终端节点策略。

Application Auto Scaling 和数据保护

Amazon 责任共担模式适用于 Application Auto Scaling 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用控制台、API、Amazon CLI 或 Amazon 软件开发工具包处理 Application Auto Scaling 或其他 Amazon 服务时。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

适用于 Application Auto Scaling 的 Identity and Access Management

Amazon Identity and Access Management (IAM) 是一种 Amazon Web Services (Amazon) 服务，可以帮助管理员安全地控制对 Amazon 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）来使用 Amazon 资源。IAM 是一个可以免费使用的 Amazon 服务。

您首先需要有一个 Amazon Web Services 账户以及用于登录账户的特定用户凭证，然后才能使用 Application Auto Scaling。要快速开始使用 Application Auto Scaling，请执行 设置 (p. 3) 中的步骤。

设置好账户后，如果要通过其 API 访问 Application Auto Scaling，无论是通过 Query (HTTPS) 接口直接访问，还是通过 SDK、Amazon Command Line Interface 或 Amazon Tools for Windows PowerShell 间接访问，都需要获取 Amazon 访问密钥。Amazon 访问密钥包含一个访问密钥 ID 和一个秘密访问密钥。有关获取 Amazon 访问密钥的更多信息，请参阅 Amazon 一般参考中的 Amazon 安全凭证。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 Application Auto Scaling 资源。例如，您必须拥有相应的权限才能执行创建扩展策略、配置计划扩展等操作。

以下部分提供了详细信息来说明 IAM 管理员如何使用 IAM 控制哪些用户可执行 Application Auto Scaling API 操作，从而对您的 Amazon 资源进行保护。

主题

Application Auto Scaling 如何与 IAM 一起使用

> **Note**
>
> 2017 年 12 月，对 Application Auto Scaling 进行了更新，同时为 Application Auto Scaling 集成服务启用了多个服务相关角色。需要特定的 IAM 权限和 Application Auto Scaling 服务相关角色（或用于 Amazon EMR 弹性伸缩的服务角色），以便用户可以配置扩缩。

在使用 IAM 管理对 Application Auto Scaling 的访问权限之前，您应该了解哪些 IAM 功能可用于 Application Auto Scaling。要大致了解 Application Auto Scaling 和其他 Amazon 服务如何与 IAM 协同工作，请参阅 IAM 用户指南中的与 IAM 协同工作的 Amazon 服务。

主题

Application Auto Scaling 基于身份的策略

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。Application Auto Scaling 支持特定的操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

操作

管理员可以使用 Amazon 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 Amazon 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相关操作的权限。

IAM policy 语句中的 Application Auto Scaling API 操作在操作前使用以下前缀：`application-autoscaling:`。策略语句必须包含 `Action` 或 `NotAction` 元素。Application Auto Scaling 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下例所示。

```
"Action": [
      "application-autoscaling:DescribeScalingPolicies",
      "application-autoscaling:DescribeScalingActivities"
```

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，请包括以下操作。

```
"Action": "application-autoscaling:Describe*"
```

要查看 Application Auto Scaling 策略操作的完整列表，请参阅《服务授权参考》中的 Application Auto Scaling 的操作、资源和条件键。

资源

`Resource` 元素指定要向其应用操作的对象。

Application Auto Scaling 没有可用作 IAM policy 语句的 `Resource` 元素的服务定义的资源。因此，IAM policy 中没有适用于 Application Auto Scaling 的 Amazon 资源名称（ARN）。要控制对 Application Auto Scaling API 操作的访问，请在编写 IAM policy 时始终使用 *（星号）作为资源。

条件键

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。例如，您可能希望策略仅在特定日期后应用。要表示条件，请使用预定义的条件键。

Application Auto Scaling 不提供任何特定于服务的条件键，但支持使用某些全局条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

`Condition` 元素是可选的。

示例

要查看 Application Auto Scaling 基于身份的策略的示例，请参阅 Application Auto Scaling 基于身份的策略示例 (p. 86)。

Application Auto Scaling 基于资源的策略

其他 Amazon 服务（如 Amazon Simple Storage Service）支持基于资源的权限策略。例如，您可以将权限策略挂载到 S3 存储桶以管理对该存储桶的访问权限。

Application Auto Scaling 不支持基于资源的策略。

访问控制列表 (ACL)

Application Auto Scaling 不支持访问控制列表 (ACL)。

基于 Application Auto Scaling 标签的授权

Application Auto Scaling 没有可添加标签的服务定义的资源。因此，它不支持基于标签控制访问。

Application Auto Scaling IAM 角色

IAM 角色是 Amazon Web Services 账户 中具有特定权限的实体。

将临时凭证与 Application Auto Scaling 一起使用

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 Amazon STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

Application Auto Scaling 支持使用临时凭证。

服务相关角色

服务相关角色向 Application Auto Scaling 授予权限，以便它可以代表您对其他 Amazon 服务进行特定调用。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Application Auto Scaling 支持服务相关角色。有关更多信息，请参阅Application Auto Scaling 的服务相关角色 (p. 83)。

服务角色

如果您的 Amazon EMR 集群使用弹性伸缩，则此功能允许 Application Auto Scaling 代表您担任服务角色。与服务相关角色类似，服务角色允许此服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Application Auto Scaling 仅支持 Amazon EMR 的服务角色。有关 EMR 服务角色的文档，请参阅 Amazon EMR Management Guide 中的 Using automatic scaling with a custom policy for instance groups。

> **Note**
>
> 引入服务相关角色之后，不再需要旧式服务角色，例如，适用于 Amazon ECS 和竞价型实例集的服务角色。

Application Auto Scaling 的 Amazon 托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管式策略。这些策略涵盖常见使用案例，可在您的 Amazon Web Services 账户 中使用。有关 Amazon 托管式策略的更多信息，请参阅 IAM 用户指南中的Amazon 托管式策略。

Amazon Web Services 负责维护和更新 Amazon 托管式策略。您无法更改 Amazon 托管式策略中的权限。服务偶尔会向 Amazon 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 Amazon 托管式策略。服务不会从 Amazon 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管式策略。例如，`ViewOnlyAccess` Amazon 托管式策略提供对许多 Amazon Web Services 服务和资源的只读访问权限。当服务启动新功能时，Amazon 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 Amazon 托管策略。

目录

Amazon 托管式策略授予对 AppStream 2.0 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingAppStreamFleetPolicy

您不能将 `AWSApplicationAutoscalingAppStreamFleetPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon AppStream 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`appstream:DescribeFleets`
- 操作：`appstream:UpdateFleet`
- 操作：`cloudwatch:DescribeAlarms`

- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Aurora 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingRDSClusterPolicy

您不能将 `AWSApplicationAutoscalingRDSClusterPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Aurora 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_RDSCluster` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`rds:AddTagsToResource`
- 操作：`rds:CreateDBInstance`
- 操作：`rds:DeleteDBInstance`
- 操作：`rds:DescribeDBClusters`
- 操作：`rds:DescribeDBInstance`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Amazon Comprehend 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingComprehendEndpointPolicy

您不能将 `AWSApplicationAutoscalingComprehendEndpointPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon Comprehend 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`comprehend:UpdateEndpoint`
- 操作：`comprehend:DescribeEndpoint`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 DynamoDB 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingDynamoDBTablePolicy

您不能将 AWSApplicationAutoscalingDynamoDBTablePolicy 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 DynamoDB 和 CloudWatch 并代表您执行扩缩。

权限详细信息

AWSServiceRoleForApplicationAutoScaling_DynamoDBTable 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：dynamodb:DescribeTable
- 操作：dynamodb:UpdateTable
- 操作：cloudwatch:DescribeAlarms
- 操作：cloudwatch:PutMetricAlarm
- 操作：cloudwatch:DeleteAlarms

Amazon 托管式策略授予对 Amazon ECS 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingECSServicePolicy

您不能将 AWSApplicationAutoscalingECSServicePolicy 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon ECS 和 CloudWatch 并代表您执行扩缩。

权限详细信息

AWSServiceRoleForApplicationAutoScaling_ECSService 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：ecs:DescribeServices
- 操作：ecs:UpdateService
- 操作：cloudwatch:DescribeAlarms
- 操作：cloudwatch:PutMetricAlarm
- 操作：cloudwatch:DeleteAlarms

Amazon 托管式策略授予对 ElastiCache 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingElastiCacheRGPolicy

您不能将 AWSApplicationAutoscalingElastiCacheRGPolicy 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 ElastiCache 和 CloudWatch 并代表您执行扩缩。

权限详细信息

AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG 服务相关角色权限策略允许 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：所有资源上的 elasticache:DescribeReplicationGroups
- 操作：所有资源上的 elasticache:ModifyReplicationGroupShardConfiguration
- 操作：所有资源上的 elasticache:IncreaseReplicaCount
- 操作：所有资源上的 elasticache:DecreaseReplicaCount

- 操作：所有资源上的 `elasticache:DescribeCacheClusters`
- 操作：所有资源上的 `elasticache:DescribeCacheParameters`
- 操作：所有资源上的 `cloudwatch:DescribeAlarms`
- 操作：资源 `arn:*:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:PutMetricAlarm`
- 操作：资源 `arn:*:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:DeleteAlarms`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Amazon Keyspaces 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingCassandraTablePolicy

您不能将 `AWSApplicationAutoscalingCassandraTablePolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon Keyspaces 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_CassandraTable` 服务相关角色权限策略允许 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system/table/*` 上的 `cassandra:Select`
- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system_schema/table/*` 上的 `cassandra:Select`
- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*` 上的 `cassandra:Select`
- 操作：资源 `arn:*:cassandra:*:*:"*"` 上的 `cassandra:Alter`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Lambda 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingLambdaConcurrencyPolicy

您不能将 `AWSApplicationAutoscalingLambdaConcurrencyPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Lambda 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`lambda:PutProvisionedConcurrencyConfig`
- 操作：`lambda:GetProvisionedConcurrencyConfig`
- 操作：`lambda:DeleteProvisionedConcurrencyConfig`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`

- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Amazon MSK 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingKafkaClusterPolicy

您不能将 `AWSApplicationAutoscalingKafkaClusterPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon MSK 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_KafkaCluster` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`kafka:DescribeCluster`
- 操作：`kafka:DescribeClusterOperation`
- 操作：`kafka:UpdateBrokerStorage`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 Neptune 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingNeptuneClusterPolicy

您不能将 `AWSApplicationAutoscalingNeptuneClusterPolicy` 附加到自己的 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Neptune 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` 服务相关角色权限策略允许 Application Auto Scaling 对指定的资源完成以下操作：

- Amazon Neptune 数据库引擎 (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}}`) 中在带有前缀 autoscaled-reader 的资源上的操作：`rds:AddTagsToResource`
- 操作：所有资源上的 `rds:ListTagsForResource`
- Amazon Neptune 数据库引擎 (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}}`) 中在所有数据库集群 (`"Resource":"arn:*:rds:*:*:db:autoscaled-reader*"`, `"arn:aws:rds:*:*:cluster:*"`) 中带有前缀 autoscaled-reader 的资源上的操作：`rds:CreateDBInstance`
- 操作：所有资源上的 `rds:DescribeDBInstances`
- 操作：所有资源上的 `rds:DescribeDBClusters`
- 操作：所有资源上的 `rds:DescribeDBClusterParameters`
- 操作：资源 `arn:*:rds:*:*:db:autoscaled-reader*` 上的 `rds:DeleteDBInstance`
- 操作：所有资源上的 `cloudwatch:DescribeAlarms`
- 操作：资源 `arn:*:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:PutMetricAlarm`

- 操作：资源 `arn:*:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:DeleteAlarms`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 SageMaker 和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingSageMakerEndpointPolicy

您不能将 `AWSApplicationAutoscalingSageMakerEndpointPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 SageMaker 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`sagemaker:DescribeEndpoint`
- 操作：`sagemaker:DescribeEndpointConfig`
- 操作：`sagemaker:UpdateEndpointWeightsAndCapacities`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对 EC2 Spot 实例集和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoscalingEC2SpotFleetRequestPolicy

您不能将 `AWSApplicationAutoscalingEC2SpotFleetRequestPolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Amazon EC2 和 CloudWatch 并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`ec2:DescribeSpotFleetRequests`
- 操作：`ec2:ModifySpotFleetRequest`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略授予对自定义资源和 CloudWatch 的访问权限

策略名称：AWSApplicationAutoScalingCustomResourcePolicy

您不能将 `AWSApplicationAutoScalingCustomResourcePolicy` 附加到 Amazon Identity and Access Management (IAM) 实体。此策略附加到服务相关角色，允许 Application Auto Scaling 通过 API Gateway和 CloudWatch 调用可用的自定义资源并代表您执行扩缩。

权限详细信息

`AWSServiceRoleForApplicationAutoScaling_CustomResource` 服务相关角色权限策略允许 Application Auto Scaling 对所有相关资源 ("Resource": "*") 完成以下操作：

- 操作：`execute-api:Invoke`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:PutMetricAlarm`
- 操作：`cloudwatch:DeleteAlarms`

Amazon 托管式策略的 Application Auto Scaling 更新

查看有关对 Amazon 托管式策略的 Application Auto Scaling 更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提醒，请订阅 Application Auto Scaling Document history（文档历史记录）页面上的 RSS 源。

更改	说明	日期
Application Auto Scaling 添加 Neptune 策略	Application Auto Scaling 为 Neptune 添加了一个新的托管式策略。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 Neptune 和 CloudWatch 并代表您执行扩缩。	2021 年 10 月 6 日
Application Auto Scaling 添加了 ElastiCache for Redis 策略	Application Auto Scaling 为 ElastiCache 添加了一个新的托管式策略。此策略附加到服务相关角色，允许 Application Auto Scaling 调用 ElastiCache 和 CloudWatch 并代表您执行扩缩。	2021 年 8 月 19 日
Application Auto Scaling 已开启跟踪更改	Application Auto Scaling 为其 Amazon 托管式策略开启了跟踪更改。	2021 年 8 月 19 日

Application Auto Scaling 的服务相关角色

Application Auto Scaling 使用服务相关角色获取代表您调用其他 Amazon 服务所需的权限。服务相关角色是一种独特类型的 Amazon Identity and Access Management（IAM）角色，它与Amazon服务直接相关。服务相关角色提供了一种将权限委托给 Amazon 服务的安全方式，因为只有相关服务才能担任服务相关角色。

目录

- 服务相关角色 ARN 参考 (p. 85)

概览

对于与 Application Auto Scaling 集成的服务，Application Auto Scaling 将为您创建服务相关角色。每个服务都有一个服务相关角色。每个服务相关角色信任指定的服务委托人来代入该角色。有关更多信息，请参阅可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7)。

Application Auto Scaling 包含每个服务相关角色的所有必要权限。这些托管式权限由 Application Auto Scaling 创建和管理，它们定义每种资源类型允许的操作。有关每个角色授予的权限的详细信息，请参阅 Application Auto Scaling 的 Amazon 托管式策略 (p. 77)。

下面的部分介绍如何创建和管理 Application Auto Scaling 服务相关角色。首先配置权限以允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。

创建服务相关角色所需的权限

您的 Amazon Web Services 账户 中的任何用户初次为指定服务调用 `RegisterScalableTarget` 时，Application Auto Scaling 需要创建服务相关角色的权限。如果服务相关角色不存在，Application Auto Scaling 会为您账户中的目标服务创建该角色。此服务相关角色向 Application Auto Scaling 授予权限，以便它能代表您调用目标服务。

为使自动角色创建成功，用户必须具有 `iam:CreateServiceLinkedRole` 操作的权限。

```
"Action": "iam:CreateServiceLinkedRole"
```

下面是允许 IAM 用户或角色为 Spot 实例集创建服务相关角色的权限策略。您可以在策略的 `Resource` 字段中将服务相关角色指定为 ARN，并将服务相关角色的服务委托人指定为条件，如下所示。有关每种服务的 ARN，请参阅 服务相关角色 ARN 参考 (p. 85)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/ec2.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName":"ec2.application-autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> `iam:AWSServiceName` IAM 条件键将指定角色附加到的服务委托人，在本示例策略中指示为 *`ec2.application-autoscaling`*`.amazonaws.com`。不要尝试猜测服务委托人。要查看服务的服务委托人，请参阅 可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7)。

创建服务相关角色（自动）

无需手动创建服务相关角色。Application Auto Scaling 将在您调用 `RegisterScalableTarget` 时为您创建相应的服务相关角色。例如，如果您已为 Amazon ECS 服务设置弹性伸缩，则 Application Auto Scaling 会创建 `AWSServiceRoleForApplicationAutoScaling_ECSService` 角色。

创建服务相关角色（手动）

要创建服务相关角色，您可以使用 IAM 控制台、Amazon CLI 或 IAM API。有关更多信息，请参阅 IAM 用户指南中的创建服务相关角色。

创建服务相关角色 (Amazon CLI)

使用以下 create-service-linked-role CLI 命令可创建 Application Auto Scaling 服务相关角色。在请求中，指定服务名称“前缀”。

要查找服务名称前缀，请参阅关于 可以与 Application Auto Scaling 一起使用的 Amazon 服务 (p. 7) 部分中每个服务的服务相关角色的服务委托人的信息。服务名称和服务委托人共享相同的前缀。例如，要创建 Amazon Lambda 服务相关角色，请使用 `lambda.application-autoscaling.amazonaws.com`。

```
aws iam create-service-linked-role --aws-service-name prefix.application-
autoscaling.amazonaws.com
```

编辑服务相关角色

对于 Application Auto Scaling 创建的服务相关角色，您只能编辑其描述。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除服务相关角色

如果您不再将 Application Auto Scaling 用于支持的服务，我们建议您删除相应的服务相关角色。

只有在先删除相关 Amazon 资源后，才能删除服务相关角色。这可以防止您无意中撤销 Application Auto Scaling 对您的资源的权限。有关更多信息，请参阅有关可扩展资源的文档。例如，要删除 Amazon ECS 服务，请参阅 Amazon Elastic Container Service 开发者指南中的删除服务。

您可以使用 IAM 删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的删除服务相关角色。

在删除某个服务相关角色后，当您调用 `RegisterScalableTarget` 时，Application Auto Scaling 将重新创建该角色。

Application Auto Scaling 服务相关角色支持的区域

Application Auto Scaling 支持在该服务可用的所有 Amazon 区域中使用服务相关角色。

服务相关角色 ARN 参考

服务	ARN
AppStream 2.0	`arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`
Aurora	`arn:aws:iam::012345678910:role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster`
Comprehend	`arn:aws:iam::012345678910:role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint`

服务	ARN
DynamoDB	arn:aws:iam::*012345678910*:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable
ECS	arn:aws:iam::*012345678910*:role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService
ElastiCache	arn:aws:iam::*012345678910*:role/aws-service-role/elasticache.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG
Keyspaces	arn:aws:iam::*012345678910*:role/aws-service-role/cassandra.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable
Lambda	arn:aws:iam::*012345678910*:role/aws-service-role/lambda.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency
MSK	arn:aws:iam::*012345678910*:role/aws-service-role/kafka.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_KafkaCluster
Neptune	arn:aws:iam::*012345678910*:role/aws-service-role/neptune.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_NeptuneCluster
SageMaker	arn:aws:iam::*012345678910*:role/aws-service-role/sagemaker.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint
Spot Fleets	arn:aws:iam::*012345678910*:role/aws-service-role/ec2.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest
自定义资源	arn:aws:iam::*012345678910*:role/aws-service-role/custom-resource.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CustomResource

Note

您可以在 Amazon CloudFormation 堆栈模板中为 AWS::ApplicationAutoScaling::ScalableTarget 资源的 `RoleARN` 属性指定服务相关角色的 ARN，即使指定的服务相关角色尚未存在。Application Auto Scaling 将自动为您创建该角色。

Application Auto Scaling 基于身份的策略示例

默认情况下，全新的 IAM 用户没有执行任何操作的权限。IAM 管理员必须创建并分配 IAM policy，以便为最终用户授予执行 Application Auto Scaling API 操作的权限。

要了解如何使用以下示例 JSON 策略文档创建 IAM policy，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

目录

- 目标资源和 CloudWatch 上 API 操作所需的权限 (p. 88)
- 在 Amazon Web Services Management Console中工作的权限 (p. 93)

Application Auto Scaling API 操作所需的权限

以下策略为调用 Application Auto Scaling API 时的常见使用案例授予权限。设置 访问控制 (p. 74) 并编写您可以附加到 IAM 用户或角色的权限策略时，请参阅本节。每个策略授予全部或部分 Application Auto Scaling API 操作的访问权限。您还需要确保 IAM 用户或角色具有目标服务和 CloudWatch 的权限策略（有关详细信息，请参阅下一节）。

以下权限策略授予全部 Application Auto Scaling API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

以下权限策略授予对配置扩缩策略而非计划操作所需的全部 Application Auto Scaling API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:RegisterScalableTarget",
              "application-autoscaling:DescribeScalableTargets",
              "application-autoscaling:DeregisterScalableTarget",
              "application-autoscaling:PutScalingPolicy",
              "application-autoscaling:DescribeScalingPolicies",
              "application-autoscaling:DescribeScalingActivities",
              "application-autoscaling:DeleteScalingPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

以下权限策略授予对配置计划操作而非扩缩策略所需的全部 Application Auto Scaling API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:RegisterScalableTarget",
              "application-autoscaling:DescribeScalableTargets",
              "application-autoscaling:DeregisterScalableTarget",
              "application-autoscaling:PutScheduledAction",
              "application-autoscaling:DescribeScheduledActions",
              "application-autoscaling:DescribeScalingActivities",
```

```
              "application-autoscaling:DeleteScheduledAction"
            ],
            "Resource": "*"
        }
    ]
}
```

目标资源和 CloudWatch 上 API 操作所需的权限

要成功配置 Application Auto Scaling 并将其与目标服务一起使用，必须授予 IAM 用户 Amazon CloudWatch 和将配置扩缩的每个目标服务所需的权限。使用以下策略为用户授予使用目标服务和 CloudWatch 所需的最低权限。

目录

AppStream 2.0 队列

以下权限策略授予对所需的所有 AppStream 2.0 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "appstream:DescribeFleets",
              "appstream:UpdateFleet",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Aurora 副本

以下权限策略授予对所需的所有 Aurora 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "rds:AddTagsToResource",
              "rds:CreateDBInstance",
              "rds:DeleteDBInstance",
              "rds:DescribeDBClusters",
              "rds:DescribeDBInstances",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Comprehend 文档分类和实体识别程序终端节点

以下权限策略授予对所需的所有 Amazon Comprehend 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "comprehend:UpdateEndpoint",
              "comprehend:DescribeEndpoint",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

DynamoDB 表和全局二级索引

以下权限策略授予对所需的所有 DynamoDB 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "dynamodb:DescribeTable",
              "dynamodb:UpdateTable",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

ECS 服务

以下权限策略授予对所需的所有 ECS 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "ecs:DescribeServices",
              "ecs:UpdateService",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

ElastiCache 复制组

以下权限策略授予对所需的所有 ElastiCache 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "elasticache:ModifyReplicationGroupShardConfiguration",
              "elasticache:IncreaseReplicaCount",
              "elasticache:DecreaseReplicaCount",
              "elasticache:DescribeReplicationGroups",
              "elasticache:DescribeCacheClusters",
              "elasticache:DescribeCacheParameters",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR 集群

以下权限策略授予对所需的所有 Amazon EMR 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "elasticmapreduce:ModifyInstanceGroups",
              "elasticmapreduce:ListInstanceGroups",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Keyspaces 表

以下权限策略授予对所需的所有 Amazon Keyspaces 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "cassandra:Select",
              "cassandra:Alter",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Lambda 函数

以下权限策略授予对所需的所有 Lambda 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "lambda:PutProvisionedConcurrencyConfig",
              "lambda:GetProvisionedConcurrencyConfig",
              "lambda:DeleteProvisionedConcurrencyConfig",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Managed Streaming for Apache Kafka (MSK) 代理存储

以下权限策略授予对所需的所有 Amazon MSK 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "kafka:DescribeCluster",
```

```
                "kafka:DescribeClusterOperation",
                "kafka:UpdateBrokerStorage",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Neptune 集群

以下权限策略授予对所需的所有 Neptune 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds:AddTagsToResource",
                "rds:CreateDBInstance",
                "rds:DescribeDBInstances",
                "rds:DescribeDBClusters",
                "rds:DescribeDBClusterParameters",
                "rds:DeleteDBInstance",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

SageMaker 终端节点

以下权限策略授予对所需的所有 SageMaker 和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sagemaker:DescribeEndpoint",
                "sagemaker:DescribeEndpointConfig",
                "sagemaker:UpdateEndpointWeightsAndCapacities",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Spot 实例集（Amazon EC2）

以下权限策略授予对所需的所有 Spot 实例集和 CloudWatch API 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "ec2:DescribeSpotFleetRequests",
              "ec2:ModifySpotFleetRequest",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

自定义资源

以下权限策略授予用户执行 API Gateway API 操作所需的权限。此策略还授予对所需的所有 CloudWatch 操作的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "execute-api:Invoke",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

在 Amazon Web Services Management Console中工作的权限

没有独立的 Application Auto Scaling 控制台。与 Application Auto Scaling 集成的大多数服务都具有专用于帮助您通过控制台配置扩缩的功能。

在大多数情况下，每个服务都提供 Amazon 托管式（预定义）IAM policy，用于定义对其控制台的访问权限，其中包括对 Application Auto Scaling API 操作的权限。有关详细信息，请参阅要使用其控制台的服务的文档。

您还可以创建自己的自定义 IAM policy，为用户授予在 Amazon Web Services Management Console 中查看和处理特定 Application Auto Scaling API 操作的精细权限。您可以使用之前章节中的策略；但是，这些策略设计用于使用 Amazon CLI 或软件开发工具包发出的请求。控制台使用其他 API 操作实现其功能，因此这些策略可能不会按预期方式起作用。例如，要配置分步扩缩，用户可能需要额外的权限来创建和管理 CloudWatch 警报。

> **Tip**
>
> 为帮助您了解在控制台中执行任务所需的相应 API 操作，您可以使用 Amazon CloudTrail 等服务。有关更多信息，请参阅 Amazon CloudTrail 用户指南。

下面显示允许用户为 Spot 队列配置扩展策略的权限策略示例。除了 Spot 实例集的 IAM 权限之外，从控制台访问实例集扩缩设置的 IAM 用户必须拥有支持动态扩缩的服务的适当权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ec2:DescribeSpotFleetRequests",
                "ec2:ModifySpotFleetRequest",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/ec2.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName":"ec2.application-autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

此策略允许用户在 Amazon EC2 控制台中查看和修改扩缩策略，并在 CloudWatch 控制台中创建和管理 CloudWatch 警报。

您可以调整 API 操作以限制用户访问权限。例如，将 `application-autoscaling:Describe*` 替换为 `application-autoscaling:*` 意味着用户具有只读访问权限。

还可以根据需要调整 CloudWatch 权限，以限制用户对 CloudWatch 功能的访问权限。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用 CloudWatch 控制台所需的权限。

Application Auto Scaling 访问故障排除

如果使用 Application Auto Scaling 时遇到 `AccessDeniedException` 或类似的困难，请参阅本节中的信息。

我无权在 Application Auto Scaling 中执行操作

如果您在调用 Amazon API 操作时收到 `AccessDeniedException`，则表明您使用的 Amazon Identity and Access Management (IAM) 用户或角色凭证没有发起该调用所需的权限。

如果 `mateojackson` IAM 用户尝试查看有关可扩展目标的详细信息，但没有 `application-autoscaling:DescribeScalableTargets` 权限，则会出现以下示例错误。

```
An error occurred (AccessDeniedException) when calling the DescribeScalableTargets
 operation: User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 application-autoscaling:DescribeScalableTargets
```

如果您收到此错误或类似错误，则必须联系您的管理员寻求帮助。

您账户的管理员需要确保您的 IAM 用户或角色具有访问 Application Auto Scaling 用于访问目标服务和 CloudWatch 中资源的所有 API 操作的权限。根据您使用的资源，需要不同的权限。用户初次配置指定资源的扩缩时，Application Auto Scaling 还需要创建服务相关角色的权限。

我是管理员并希望允许其他人访问 Application Auto Scaling

要允许其他人访问 Application Auto Scaling，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 Application Auto Scaling 中向其授予正确的权限。

要开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我是管理员，我的 IAM policy 返回错误或未按预期工作

除了 Application Auto Scaling API 操作所需的 IAM 权限外，您的 IAM 权限策略还必须授予调用目标服务和 CloudWatch 的访问权限。

如果用户或应用程序没有适当的 IAM policy 权限，其访问可能会被意外拒绝。要为账户中的用户和应用程序编写权限策略，请参阅 Application Auto Scaling 基于身份的策略示例 (p. 86) 中的信息。

有关如何执行验证的信息，请参阅 对目标资源进行 API 调用的权限验证 (p. 95)。

请注意，某些权限问题也可能是由于创建 Application Auto Scaling 所使用的服务相关角色时出现问题所致。有关创建这些服务相关角色的信息，请参阅 Application Auto Scaling 的服务相关角色 (p. 83)。

对目标资源进行 API 调用的权限验证

向 Application Auto Scaling API 操作发出授权请求需要 API 调用方具有访问目标服务和 CloudWatch 中的 Amazon 资源的权限。Application Auto Scaling 会验证与目标服务和 CloudWatch 关联的请求的权限，然后再继续处理请求。为此，我们将发出一系列调用来验证目标资源的 IAM 权限。返回响应时，Application Auto Scaling 会读取该响应。如果 IAM 权限不允许指定的操作，则 Application Auto Scaling 将使请求失败，并将错误返回给用户，其中包含有关缺少权限的信息。这可确保用户想要部署的扩缩配置按预期工作，并且在请求失败时返回有用的错误。

作为工作原理的示例，以下信息提供了有关 Application Auto Scaling 如何通过 Aurora 和 CloudWatch 执行权限验证的详细信息。

当 IAM 用户针对 Aurora 数据库集群调用 `RegisterScalableTarget` API 时，Application Auto Scaling 会执行以下所有检查以验证 IAM 用户是否具有所需的权限（以粗体显示）。

- rds:CreateDBInstance：为确定用户是否具有此权限，我们将向 `CreateDBInstance` API 操作发送请求，尝试在用户指定的 Aurora 数据库集群中创建具有无效参数（空实例 ID）的数据库实例。对于授权用户，该 API 将在审计请求后返回 `InvalidParameterValue` 错误代码响应。但是，对于未经授权的用户，我们会收到 `AccessDenied` 错误，使 Application Auto Scaling 请求失败并显示 `ValidationException` 错误，向用户列出缺少的权限。
- rds:DeleteDBInstance：我们将向 `DeleteDBInstance` API 操作发出一个空实例 ID。对于授权用户，此请求会导致 `InvalidParameterValue` 错误。对于未经授权的用户，它会导致 `AccessDenied` 并向用户发送验证异常（与第一个要点中描述的处理相同）。
- rds:AddTagsToResource：由于 `AddTagsToResource` API 操作需要 Amazon Resource Name (ARN)，因此必须使用无效的账户 ID (12345) 和虚拟实例 ID (non-existing-db) 指定一个“虚拟”资源以构建 ARN (`arn:aws:rds:us-east-1:12345:db:non-existing-db`)。对于授权用户，此请求会导致

`InvalidParameterValue` 错误。对于未经授权的用户，它会导致 `AccessDenied` 并向用户发送验证异常。

- rds:DescribeDBCluster：我们描述为弹性伸缩注册的资源的集群名称。对于授权用户，我们将得到一个有效的描述结果。对于未经授权的用户，它会导致 `AccessDenied` 并向用户发送验证异常。
- rds:DescribeDBInstance。我们使用 `db-cluster-id` 筛选条件调用 `DescribeDBInstance` API，筛选用户提供的集群名称以注册可扩展目标。对于授权用户，我们可以描述数据库集群中的所有数据库实例。对于未经授权的用户，此调用会导致 `AccessDenied` 并向用户发送验证异常。
- cloudwatch:PutMetricAlarm：我们不带任何参数调用 `PutMetricAlarm` API。由于缺少警报名称，对于授权用户，请求会导致 `ValidationError`。对于未经授权的用户，它会导致 `AccessDenied` 并向用户发送验证异常。
- cloudwatch:DescribeAlarms：我们调用 `DescribeAlarms` API 并将最大记录数的值设置为 1。对于授权用户，我们预期响应中有一个警报的信息。对于未经授权的用户，此调用会导致 `AccessDenied` 并向用户发送验证异常。
- cloudwatch:DeleteAlarms：与上面的 `PutMetricAlarm` 类似，但我们不为 `DeleteAlarms` 请求提供参数。由于请求中缺少警报名称，对于授权用户，此调用将失败并显示 `ValidationError`。对于未经授权的用户，它会导致 `AccessDenied` 并向用户发送验证异常。

只要发生任何一个验证异常，它就会被记录下来。您可以通过使用 Amazon CloudTrail，采取步骤手动标识哪些调用验证失败。有关更多信息，请参阅 Amazon CloudTrail 用户指南。

Note

如果您在使用 CloudTrail 时收到应用程序弹性伸缩提示，则默认情况下，这些提示将包括用于验证用户权限的 Application Auto Scaling 调用。要过滤掉这些提示，请使用 `invokedBy` 字段，它们包含用于这些验证检查的 `application-autoscaling.amazonaws.com`。

Application Auto Scaling 的合规性验证

作为多个 Amazon 合规性计划的一部分，第三方审核员将评估 Amazon Web Services的安全性与合规性，例如 SOC、PCI、FedRAMP 和 HIPAA。

要了解此服务或其他 Amazon Web Services 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 Amazon Web Services。有关常规信息，请参阅Amazon Web Services合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅、在 Amazon Artifact 中下载报告。

您使用 Amazon Web Services 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。Amazon 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Amazon上部署注重安全性和合规性的基准环境的步骤。
- 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 Amazon Web Services 创建符合 HIPAA 标准的应用程序。

  Note

  并非所有 Amazon Web Services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。
- Amazon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。
- Amazon Config 开发人员指南中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub：此 Amazon Web Service 提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践规范。

Application Auto Scaling 中的恢复功能

Amazon全球基础设施围绕Amazon区域和可用区构建。

Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。

利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

Application Auto Scaling 中的基础设施安全性

作为一项托管式服务，Application Auto Scaling 由 Amazon Web Services：安全流程概览白皮书中所述的 Amazon 全球网络安全流程提供保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 Application Auto Scaling。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。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 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

Application Auto Scaling 配额

您的 Amazon Web Services 账户 对于每项 Amazon 服务都具有默认配额（以前称为限制）。除非另有说明，否则，每个配额是区域特定的。您可以请求增加某些配额，但其他一些配额无法增加。

要查看 Application Auto Scaling 配额，请打开 Service Quotas 控制台。在导航窗格中，选择 Amazon services（Amazon 服务），然后选择 Application Auto Scaling。

要请求提高配额，请参阅 Service Quotas 用户指南中的请求提高配额。如果配额在 Service Quotas 中尚不可用，请使用 Application Auto Scaling 限制表。确保在增加请求中指定资源的类型，例如 Amazon ECS 或 DynamoDB。

您的 Amazon Web Services 账户 具有以下 Application Auto Scaling 相关配额。

每个账户每个区域的默认配额

项目	默认值	可调整
每个资源类型的最大可扩展目标数	默认配额因资源类型而异。 对于所有其他资源类型的每一个，最多为 5000 个 Amazon DynamoDB 可扩展目标、3000 个 ECS 可扩展目标和 500 个可扩展目标。	是
每个可扩展目标的最大扩展策略数	50 包含步进扩展策略和目标跟踪策略。	否
每个可扩展目标的最大计划操作数	200	否
每个步进扩展策略的最大步进调整数	20	否

在扩展工作负载时，请牢记服务配额。例如，当您达到某个服务允许的最大容量单位数时，向外扩展操作将会停止。如果需求下降并且当前容量下降，则 Application Auto Scaling 会再次横向扩展。为避免再次达到此容量限制，您可以请求增加配额限制。对于最大资源容量，每个服务都有各自的默认配额。有关其他 Amazon 服务默认配额的信息，请参阅 Amazon Web Services 一般参考中的服务终端节点和配额。

使用 Amazon CloudFormation 创建 Application Auto Scaling 资源

Application Auto Scaling 与 Amazon CloudFormation 集成，后者是一项服务，可帮助您对 Amazon 资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个描述所需的全部 Amazon 资源的模板，Amazon CloudFormation 将为您预置和配置这些资源。

在您使用 Amazon CloudFormation 时，可重复使用您的模板来不断地重复设置您的 Application Auto Scaling 资源。描述您的资源一次，然后在多个 Amazon Web Services 账户 和区域中反复预置相同的资源。

Application Auto Scaling 和 Amazon CloudFormation 模板

要为 Application Auto Scaling 和相关服务预置和配置资源，您必须了解 Amazon CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 Amazon CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 Amazon CloudFormation Designer 的帮助下开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的什么是 Amazon CloudFormation Designer？。

为 Application Auto Scaling 资源创建堆栈模板时，必须提供以下内容：

- 目标服务的命名空间（例如 **appstream**）。请参阅 AWS::ApplicationAutoScaling::ScalableTarget 参考以获取服务命名空间。
- 与目标资源关联的可扩展维度（例如 **appstream:fleet:DesiredCapacity**）。请参阅 AWS::ApplicationAutoScaling::ScalableTarget 参考以获取可扩展维度。
- 目标资源的资源 ID（例如 **fleet/sample-fleet**）。有关特定资源 ID 的语法和示例的信息，请参阅 AWS::ApplicationAutoScaling::ScalableTarget 参考。
- 目标资源的服务相关角色（例如 **arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet**）。请参阅 服务相关角色 ARN 参考 (p. 85) 表以获取角色 ARN。

要了解有关 Application Auto Scaling 资源的更多信息，请参阅 Amazon CloudFormation 用户指南中的 Application Auto Scaling 参考。

示例模板代码段

我们提供了一些 JSON 和 YAML 模板代码段，您可以用其了解如何在堆栈模板中声明各种扩缩策略和计划的操作。有关更多信息，请参阅 Amazon CloudFormation 用户指南中的 Application Auto Scaling 模板示例部分。有关更多信息，请参阅 Amazon CloudFormation 用户指南中 Application Auto Scaling 参考中的示例部分。

了解有关 Amazon CloudFormation 的更多信息

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

- Amazon CloudFormation
- Amazon CloudFormation 用户指南
- Amazon CloudFormation API 参考
- Amazon CloudFormation 命令行界面用户指南

文档历史记录

下表介绍了自 2018 年 1 月以来对 Application Auto Scaling 文档的重要补充。如需对此文档更新的通知，您可以订阅 RSS 源。

update-history-change	update-history-description	update-history-date
指南更改 (p. 101)	更新了配额文档中的每个资源类型的最大可扩展目标数。请参阅 Application Auto Scaling 配额。	2022 年 5 月 6 日
添加对 Amazon Neptune 集群的支持 (p. 101)	使用 Application Auto Scaling 来扩展 Amazon Neptune 数据库集群中的副本数量。有关更多信息，请参阅 Amazon Neptune 和 Application Auto Scaling。主题对 Amazon 托管策略的 Application Auto Scaling 更新已更新以列出与 Neptune 集成的新托管策略。	2021 年 10 月 6 日
指南更改 (p. 101)	Application Auto Scaling 用户指南中的新 IAM 主题可帮助您解决访问 Application Auto Scaling 的问题。有关更多信息，请参阅 Identity and Access Management for Application Auto Scaling。还为对目标服务和 Amazon CloudWatch 的操作添加了新的 IAM 权限策略示例。有关更多信息，请参阅有关使用 Amazon CLI 或软件开发工具包的示例策略。	2021 年 2 月 23 日
添加对本地时区的支持 (p. 101)	现在，您可以在本地时区创建计划的操作。如果您的时区遵守夏令时，它会自动调整夏令时 (DST)。有关更多信息，请参阅计划的扩缩。	2021 年 2 月 2 日
指南更改 (p. 101)	Application Auto Scaling 用户指南中的新教程可帮助您了解在使用 Application Auto Scaling 时如何使用目标跟踪扩缩策略和计划的扩缩来提高应用程序的可用性。此外，新主题说明 CloudWatch 检测到可能需要关注的任何问题时如何触发通知。	2020 年 10 月 15 日
添加对 Amazon Managed Streaming for Apache Kafka 集群存储的支持 (p. 101)	使用目标跟踪扩缩策略以横向扩展与 Amazon MSK 集群关联的代理存储量。	2020 年 9 月 30 日
添加对 Amazon Comprehend 实体识别程序终端节点的支持 (p. 101)	使用 Application Auto Scaling 可扩展为 Amazon Comprehend 实体识别程序终端节点预置的推理单位数量。	2020 年 9 月 28 日

添加对 Amazon Keyspaces (for Apache Cassandra) 表的支持 (p. 101)	使用 Application Auto Scaling 扩展 Amazon Keyspaces 表的预置吞吐量（读取和写入容量）。	2020 年 4 月 23 日
新增“安全性”章节 (p. 101)	Application Auto Scaling 用户指南中新的安全章节可帮助您了解如何在使用 Application Auto Scaling 时应用责任共担模式。作为此更新的一部分，已将用户指南的“身份验证和访问控制”一章替换为一个新的、更实用的部分，即 Identity and Access Management for Application Auto Scaling。	2020 年 1 月 16 日
次要更新 (p. 101)	各种改进和更正。	2020 年 1 月 15 日
增加了通知功能 (p. 101)	Application Auto Scaling 现在会在某些操作发生时向 Amazon EventBridge 发送事件并向 Amazon Health Dashboard 发送通知。有关更多信息，请参阅 Application Auto Scaling 监控。	2019 年 12 月 20 日
添加对 Amazon Lambda 函数的支持 (p. 101)	使用 Application Auto Scaling 扩展 Lambda 函数的预置并发。	2019 年 12 月 3 日
添加对 Amazon Comprehend 文档分类终端节点的支持 (p. 101)	使用 Application Auto Scaling 扩展 Amazon Comprehend 文档分类终端节点的吞吐量。	2019 年 11 月 25 日
添加对目标跟踪扩缩策略的 AppStream 2.0 支持 (p. 101)	使用目标跟踪扩缩策略来扩展 AppStream 2.0 队列的规模。	2019 年 11 月 25 日
对 Amazon VPC 终端节点的支持 (p. 101)	您现在可以在 VPC 和 Application Auto Scaling 之间建立私有连接。有关迁移注意事项和说明，请参阅 Application Auto Scaling 和接口 VPC 终端节点。	2019 年 11 月 22 日
暂停和恢复扩缩 (p. 101)	增加了对暂停和恢复扩展的支持。有关更多信息，请参阅暂停和恢复 Application Auto Scaling 的扩缩。	2019 年 8 月 29 日
新章节 (p. 101)	设置部分已添加到 Application Auto Scaling 文档。对整个用户指南进行了少量改进和修复。	2019 年 6 月 28 日
指南更改 (p. 101)	改进了 Application Auto Scaling 文档中的计划的扩展、分步扩缩策略和目标跟踪扩缩策略部分。	2019 年 3 月 11 日
添加对自定义资源的支持 (p. 101)	使用 Application Auto Scaling 扩展由您自己的应用程序或服务提供的自定义资源。有关更多信息，请参阅我们的 GitHub 存储库。	2018 年 7 月 9 日
添加对 SageMaker 终端节点变体的支持 (p. 101)	使用 Application Auto Scaling 扩展为变体预置的终端节点实例数。	2018 年 2 月 28 日

下表介绍了 2018 年 1 月之前对 Application Auto Scaling 文档的重要更改。

更改	说明	日期
添加对 Aurora 副本的支持	使用 Application Auto Scaling 扩展所需的计数。有关更多信息，请参阅 Amazon RDS 用户指南中的将 Amazon Aurora Auto Scaling 与 Aurora 副本一起使用。	2017 年 11 月 17 日
添加对计划扩展的支持	使用计划的扩展在特定预设时间或按照特定预设间隔扩展资源。有关更多信息，请参阅 Application Auto Scaling 的计划扩缩。	2017 年 11 月 8 日
添加对目标跟踪扩展策略的支持	使用目标跟踪扩展策略通过几个简单步骤为您的应用程序设置动态扩展。有关更多信息，请参阅 Application Auto Scaling 的目标跟踪扩缩策略。	2017 年 12 月 7 日
添加对 DynamoDB 表和全局二级索引的预置读取和写入容量的支持	使用 Application Auto Scaling 扩展预置吞吐量（读取和写入容量）。有关更多信息，请参阅 Amazon DynamoDB 开发人员指南中的使用 DynamoDB Auto Scaling 管理吞吐量。	2017 年 6 月 14 日
添加对 AppStream 2.0 队列的支持	使用 Application Auto Scaling 扩展队列的规模。有关更多信息，请参阅 Amazon AppStream 2.0 管理指南中的适用于 AppStream 2.0 的队列 Auto Scaling。	2017 年 3 月 23 日
添加对 Amazon EMR 集群的支持	使用 Application Auto Scaling 扩展核心和任务节点。有关更多信息，请参阅 Amazon EMR 管理指南 中的在 Amazon EMR 中使用弹性伸缩。	2016 年 11 月 18 日
增加对 Spot 队列的支持	使用 Application Auto Scaling 扩展目标容量。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Spot 实例集的弹性伸缩。	2016 年 9 月 1 日
添加对 Amazon ECS 服务的支持	使用 Application Auto Scaling 扩展所需的计数。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。	2016 年 8 月 9 日