aws

用户指南

Application Auto Scaling

Application Auto Scaling: 用户指南

Table of Contents

什么是 Application Auto Scaling？

Application Auto Scaling 是一项网络服务，适用于需要一种解决方案来自动扩展其可扩展资源，用于超出 Amazon A EC2 uto Scaling 的各项 AWS 服务。使用 Application Auto Scaling，您可以为以下资源配置自动缩放：

- WorkSpaces 应用程序舰队
- Aurora 副本
- Amazon Comprehend 文档分类和实体识别程序终端节点
- DynamoDB 表和全局二级索引
- Amazon ECS 服务
- ElastiCache 复制组（Redis OSS 和 Valkey）和 Memcached 集群
- 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 AI 端点变体
- SageMaker AI 推理组件
- SageMaker AI 无服务器预配置并发
- Spot 队列请求
- 亚马逊资源池 WorkSpaces
- 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何 AWS 服务的区域可用性，请参阅区域表 。

有关使用 Auto Scaling 组扩展您的亚马逊 EC2 实例队列的信息，请参阅 Amazon A EC2 uto 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。

- 目标跟踪扩展-根据特定 CloudWatch 指标的目标值扩展资源。

- 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。
- 预测性扩展 — 根据历史数据主动扩展资源以匹配预期负载。

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：

- AWS 管理控制台 – 提供可用于配置扩缩的 Web 界面。注册一个 AWS 帐户并登录 AWS 管理控制台. 然后，打开服务控制台以查看简介中列出的资源之一。例如，要扩展 Lambda 函数，请打开。AWS Lambda console确保以与要使用的资源 AWS 区域 相同的方式打开控制台。

  > **Note**
  >
  > 并非所有资源都可以访问控制台。有关更多信息，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用。

- AWS Command Line Interface (AWS CLI) — 为大量用户提供命令，并在 Windows AWS 服务、macOS 和 Linux 上受支持。要开始使用，请参阅AWS Command Line Interface。有关更多信息，请参阅《AWS CLI 命令参考》中的 application-autoscaling。
- AWS Tools for Windows PowerShell— 为那些在 PowerShell 环境中编写脚本的用户提供一系列 AWS 产品的命令。要开始使用，请参阅《AWS Tools for PowerShell 用户指南》。有关更多信息，请参阅 AWS Tools for PowerShell Cmdlet 参考。
- AWS SDKs— 提供特定于语言的 API 操作并处理许多连接细节，例如计算签名、处理请求重试和处理错误。有关更多信息，请参阅构建工具 AWS。
- HTTPS API – 提供了您使用 HTTPS 请求调用的低级别 API 操作。有关更多信息，请参阅 Application Auto Scaling API 参考。
- CloudFormation— 支持使用 CloudFormation 模板配置缩放。有关更多信息，请参阅 使用配置应用程序 Auto Scaling 资源 AWS CloudFormation。

要以编程方式连接到 AWS 服务，请使用终端节点。有关调用 。

Application Auto Scaling 概念

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底层资源进行 out-of-band更改，而 Application Auto Scaling 对此一无所知。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

扩展策略指示 Application Auto Scaling 跟踪特定 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 auto Scaling 的指标由目标服务发布，但您也可以将自己的指标发布到扩展策略，CloudWatch 然后将其与扩展策略一起使用。

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

AWS 服务 可以与 Application Auto Scaling 一起使用 - 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 - 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 会根据需要启动或终止 EC2 实例，将所有服务器的聚合 CPU 利用率保持在 50%。

AWS 服务 可以与 Application Auto Scaling 一起使用

Application Auto Scaling 与其他 AWS 服务集成，因此您可以添加扩展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。

下表列出了可以与 Application Auto Scaling 配合使用的 AWS 服务，包括有关支持配置自动缩放的方法的信息。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 AWS 服务以启动弹性伸缩。
- CLI 访问 - 您可以配置兼容的 AWS 服务，以使用 AWS CLI启动弹性伸缩。
- SDK 访问权限 — 您可以使用配置兼容 AWS 服务以启动 auto Scaling AWS SDKs。
- CloudFormation ac@@ ces s — 您可以使用 CloudFormation 堆栈模板将兼容的 AWS 服务配置为启动自动扩展。有关更多信息，请参阅 使用配置应用程序 Auto Scaling 资源 AWS CloudFormation。

AWS 服务	控制台访问权限[1]	CLI 访问	软件开发工具包访问	CloudFormation 访问
WorkSpaces 应用程序	是	是	是	是
Aurora	是	是	是	是
Amazon Comprehend	否	是	是	是
Amazon DynamoDB	是	是	是	是

AWS 服务	控制台访问权限[1]	CLI 访问	软件开发工具包访问	CloudFormation 访问
Amazon ECS	是	是	是	是
Amazon ElastiCache	是	是	是	是
Amazon EMR	是	是	是	是
Amazon Keyspaces	是	是	是	是
Lambda	否	是	是	是
Amazon MSK	是	是	是	是
Amazon Neptune	否	是	是	是
SageMaker AI	是	是	是	是

AWS 服务	控制台访问权限[1]	CLI 访问	软件开发工具包访问	CloudFormation 访问
竞价型实例集	 是	 是	 是	 是
WorkSpaces	 是	 是	 是	 是
自定义资源	 否	 是	 是	 是

[1] 用于配置扩展策略的控制台访问权限。大多数服务不支持从控制台配置计划扩展。目前，只有 Amazon WorkSpaces 应用程序和 Spot 队列为计划扩展提供控制台访问权限。 ElastiCache

Amazon WorkSpaces 应用程序和应用程序 Auto Scaling

您可以使用目标跟踪扩展策略、步进扩展策略和计划扩展来扩展 WorkSpaces 应用程序队列。

使用以下信息来帮助您将 WorkSpaces 应用程序与应用程序 Auto Scaling 集成。

为应用程序创建的 WorkSpaces 服务相关角色

使用 Application Auto Scaling 将 WorkSpaces 应用程序资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 WorkSpaces 应用程序队列注册为可扩展目标

Application Auto Scaling 需要一个可扩展的目标，然后才能为 WorkSpaces 应用程序队列创建扩展策略或计划操作。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 WorkSpaces 应用程序控制台配置 auto scaling，则 WorkSpaces 应用程序会自动为您注册可扩展目标。

如果要使用 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS CLI:

  调用 WorkSpaces 应用程序队列的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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅《亚马逊 WorkSpaces 应用程序管理指南》中的 Fleet Auto Scaling fo WorkSpaces r Amazon 应用程序。

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。

为 Aurora 创建的服务相关角色

在 Application Auto Scaling 中将 Aurora 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS 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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅亚马逊 RDS Aurora 用户指南中的 Amazon Aurora Auto Scaling 和 Aurora 副本。

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。

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 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个配置自动缩放 AWS SDKs，可以使用以下选项：

- AWS CLI:

  为文档分类终端节点调用 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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅 Amazon Comprehend 开发者指南中的使用终端节点自动扩展。

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。

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 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS CLI:

  调用register-scalable-target命令获取表的写入容量。以下示例注册了名为的表的预配置写入容量`my-table`，该表的最小写入容量为五个写入容量单位，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
  ```

  如果成功，此命令将返回可扩展目标的 ARN：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

调用register-scalable-target命令获取表的读取容量。以下示例注册了名为的表的预配置读取容量my-table，该表的最小容量为 5 个读取容量单位，最大容量为 10 个读取单元：

```
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
```

如果成功，此命令将返回可扩展目标的 ARN：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

调用register-scalable-target命令获取全局二级索引的写入容量。以下示例注册了名为的全局二级索引的预配置写入容量my-table-index，该索引的最小写入容量为五个写入容量单位，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
```

如果成功，此命令将返回可扩展目标的 ARN：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
```

```
}
```

调用register-scalable-target命令获取全局二级索引的读取容量。以下示例注册了名为的全局二级索引的预配置读取容量my-table-index，该索引的最小容量为 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
```

如果成功，此命令将返回可扩展目标的 ARN：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

如果您刚刚开始使用 Application Auto Scaling，则可以在以下文档中找到有关扩展 DynamoDB 资源的其它有用信息：

- Amazon DynamoDB 开发人员指南中的使用 DynamoDB Auto Scaling 管理吞吐量
- 《Amazon DynamoDB 开发人员指南》中的评估表的自动扩缩设置
- CloudFormation 如何使用博客上的 DynamoDB 表和索引配置自动缩放 AWS

Amazon ECS 和 Application Auto Scaling

您可以使用目标跟踪扩展策略、预测性扩展策略、步进扩展策略和计划扩展来扩展 ECS 服务。

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

为 Amazon ECS 创建的服务相关角色

在 Application Auto Scaling 中将 Amazon ECS 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS 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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

如果您刚刚开始使用 Application Auto Scaling，则可以在以下文档中找到有关扩展 Amazon ECS 资源的其它有用信息：

- 《Amazon Elastic Container Service 开发人员指南》中的服务自动扩缩
- 在@@ 亚马逊弹性容器服务开发者指南中优化 Amazon ECS 服务的自动扩展

> **Note**
>
> 有关在 Amazon ECS 部署正在进行时暂停横向扩展过程的说明，请参阅以下文档：《Amazon Elastic Container Service 开发人员指南》中的服务自动扩缩和部署

ElastiCache 和应用程序 Auto Scaling

您可以使用目标跟踪扩展策略和计划扩展，横向扩展亚马逊 ElastiCache 复制组（Redis OSS 和 Valkey）和 Memcached 自行设计的集群。

要 ElastiCache 与 Application Auto Scaling 集成，请使用以下信息。

为创建的服务相关角色 ElastiCache

使用 Application Auto Scaling 将 ElastiCache 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 资源注册为可扩展目标

Application Auto Scaling 需要一个可扩展的目标，然后才能为 ElastiCache 复制组、集群或节点创建扩展策略或计划操作。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 auto Scaling，则 ElastiCache 会自动为您注册可扩展目标。

如果要使用 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS CLI:

  调用 ElastiCache复制组的register-scalable-target命令。以下示例注册名为 `mycluster1` 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/mycluster1 \
   --min-capacity 1 \
   --max-capacity 5
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

以下示例为名为的复制组注册每个节点组所需的副本数量mycluster2，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/mycluster2 \
   --min-capacity 1 \
   --max-capacity 5
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/234abcd56ab78cd901ef1234567890ab1234"
}
```

以下示例为名为的集群注册了所需的节点数量mynode1，其最小容量为 20，最大容量为 50。

```
aws application-autoscaling register-scalable-target \
   --service-namespace elasticache \
   --scalable-dimension elasticache:cache-cluster:Nodes \
   --resource-id cache-cluster/mynode1 \
   --min-capacity 20 \
   --max-capacity 50
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/01234abcd56ab78cd901ef1234567890ab12"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅亚马逊 ElastiCache 用户指南中的 Auto Scaling Valkey 和 Redis OSS 集群以及扩展 Memcached 集群。

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。

为 Amazon Keyspaces 创建的服务相关角色

在 Application Auto Scaling 中将 Amazon Keyspaces 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS 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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

  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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅《亚马逊密钥空间开发者指南》中的 “使用 Amazon Keysp aces 自动缩放自动管理吞吐容量”。

AWS Lambda 和应用程序 Auto Scaling

您可以使用目标跟踪扩展策略和计划扩展来扩展 AWS Lambda 预配置的并发量。

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

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 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个配置自动缩放 AWS SDKs，可以使用以下选项：

- AWS 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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

如果您刚刚开始使用 Application Auto Scaling，则可以在以下文档中找到有关扩展 Lambda 函数的其它有用信息：

- 《AWS Lambda 开发人员指南》中的配置预置并发
- 安排 Lambda 预配置并发以应对博客上反复出现的高峰使用量 AWS

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。

为 Amazon MSK 创建的服务相关角色

在 Application Auto Scaling 中将 Amazon MSK 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS 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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅《适用于 A pache Kafka 的亚马逊托管流媒体 Kafka 开发者指南》中的 Amazon MSK 集群的自动扩展。

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。

为 Neptune 创建的服务关联角色

在 Application Auto Scaling 中将 Neptune 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个配置自动缩放 AWS SDKs，可以使用以下选项：

- AWS CLI:

  调用 Neptune 集群的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
  ```

  如果成功，该命令会返回可扩展目标的 ARN。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS 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。

相关资源

有关更多信息，请参阅 Neptune 用户指南中的自动扩展 Amazon Neptune 数据库集群中的副本数量。

Amazon SageMaker AI 和应用程序 Auto Scaling

您可以使用目标跟踪扩展策略、步进扩展策略和计划扩展来扩展 SageMaker AI 终端节点变体、无服务器终端节点的预配置并发以及推理组件。

使用以下信息来帮助你将 SageMaker AI 与 Application Auto Scaling 集成。

为 AI 创建的 SageMaker 服务相关角色

使用 Application Auto Scaling 将 SageMaker AI 资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AI 端点变体注册为可扩展目标

Application Auto Scaling 需要一个可扩展的目标，然后才能为 SageMaker AI 模型（变体）创建扩展策略或计划操作。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 AI 控制台配置 auto Scaling，则 SageMaker AI 会自动为您注册可扩展目标。

如果要使用 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS CLI:

  调用register-scalable-target命令获取产品变体。以下示例为名为 `my-variant` 的产品变体（在 `my-endpoint` 端点上运行）注册所需的实例计数，最小容量为一个实例，最大容量为八个实例。

```
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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

使用 Application Auto Scaling 将无服务器端点的预置并发注册为可扩展目标

Application Auto Scaling 也需要一个可扩展目标，然后才能为无服务器端点预置并发创建扩缩策略或计划的操作。

如果您使用 SageMaker AI 控制台配置 auto Scaling，则 SageMaker AI 会自动为您注册可扩展目标。

如果没有自动注册，请使用以下方法之一注册可扩展目标：

- AWS CLI:

  调用register-scalable-target命令获取产品变体。以下示例为名为 `my-variant` 的产品变体（在 `my-endpoint` 端点上运行）注册预置并发，最小容量为一个实例，最大容量为十个实例。

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --scalable-dimension sagemaker:variant:DesiredProvisionedConcurrency \
  --resource-id endpoint/my-endpoint/variant/my-variant \
  --min-capacity 1 \
  --max-capacity 10
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

使用 Application Auto Scaling 将推理组件注册为可扩展目标

Application Auto Scaling 也需要一个可扩展目标，然后才能为推理组件创建扩展策略或计划的操作。

- AWS CLI:

  调用推理组件的register-scalable-target命令。以下示例为名为 `my-inference-component` 的推理组件注册所需的副本计数，最小容量为零个副本，最大容量为三个副本。

```
aws application-autoscaling register-scalable-target \
  --service-namespace sagemaker \
  --scalable-dimension sagemaker:inference-component:DesiredCopyCount \
  --resource-id inference-component/my-inference-component \
  --min-capacity 0 \
  --max-capacity 3
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

如果您刚刚开始使用 Application Auto Scaling，可以在亚马逊 SageMaker 人工智能开发者指南中找到有关扩展 A SageMaker I 资源的其他有用信息：

- 自动缩放 Amazon SageMaker 人工智能模型
- Automatically scale Provisioned Concurrency for a serverless endpoint
- Set auto scaling policies for multi-model endpoint deployments
- Autoscale an asynchronous endpoint

> **Note**
>
> 2023 年， SageMaker AI 推出了基于实时推理端点的新推理功能。您可以使用终端节点配置创建 A SageMaker I 终端节点，该端点配置定义了终端节点的实例类型和初始实例数。然后，创建一个推理组件，它是一个 SageMaker AI 托管对象，可用于将模型部署到终端节点。有关扩展推理组件的信息，请参阅博客上的 Amazon SageMaker AI 添加了新的推理功能以帮助降低基础模型部署成本和延迟，以及使用 Amazon A SageMaker I 的最新功能将模型部署成本平均降低 50%。 AWS

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。

为 Spot 实例集创建的服务相关角色

在 Application Auto Scaling 中将 Spot 队列资源注册为可扩展目标 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS 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
```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

有关更多信息，请参阅 Amazon EC2 用户指南中的了解竞价型队列的自动扩展。

亚马逊 WorkSpaces 和 Application Auto Scaling

WorkSpaces 您可以使用目标跟踪扩展策略、步进扩展策略和计划扩展来扩展池。

使用以下信息来帮助您 WorkSpaces 与 Application Auto Scaling 集成。

为创建的服务相关角色 WorkSpaces

当您向 Application Auto Scaling 注册 WorkSpaces 资源作为可扩展目标 AWS 账户 时，Application Auto Scaling 会自动创建AWSServiceRoleForApplicationAutoScaling_WorkSpacesPool在您的中命名的服务相关角色。有关更多信息，请参阅 Application Auto Scaling 的服务相关角色。

此服务相关角色使用托管策略 AWSApplicationAutoscalingWorkSpacesPoolPolicy。此政策授予 Application Auto Scaling WorkSpaces 代表您致电亚马逊的权限。有关更多信息，请参阅AWSApplicationAutoscalingWorkSpacesPoolPolicy《AWS 托管策略参考》。

服务相关角色使用的服务委托人

服务相关角色信任以下服务主体来代入该角色：

- `workspaces.application-autoscaling.amazonaws.com`

使用 Application Auto Scaling 将 WorkSpaces 池注册为可扩展目标

Application Auto Scaling 需要一个可扩展的目标，然后才能为其创建扩展策略或计划操作 WorkSpaces。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 WorkSpaces 控制台配置 auto Scaling，则 WorkSpaces 会自动为您注册可扩展目标。

如果要使用 AWS CLI 或其中一个来配置 auto Scaling AWS SDKs，则可以使用以下选项：

- AWS CLI:

  调用register-scalable-target命令获取一个池 WorkSpaces。以下示例 WorkSpaces 使用其请求 ID 注册池的目标容量，最小容量为两个虚拟桌面，最大容量为十个虚拟桌面。

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace workspaces \
    --resource-id workspacespool/wspool-abcdef012 \
    --scalable-dimension workspaces:workspacespool:DesiredUserSessions \
    --min-capacity 2 \
    --max-capacity 10
  ```

  如果成功，该命令会返回可扩展目标的 ARN。

  ```
  {
      "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
  }
  ```

- AWS 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。

相关资源

有关更多信息，请参阅《Amazon WorkSpaces 管理指南》中的 A uto Scaling for P WorkSpaces oo ls。

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。

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 AWS 账户 时，将在您的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。

- `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 AWS CLI 或其中一个配置自动缩放 AWS SDKs，可以使用以下选项：

- AWS 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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

- AWS 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。

相关资源

如果您刚刚开始使用 Application Auto Scaling，则可以在以下文档中找到有关扩展自定义资源的其它有用信息：

GitHub存储库

使用配置应用程序 Auto Scaling 资源 AWS CloudFormation

Application Auto Scaling 与 AWS CloudFormation一项服务集成，可帮助您对 AWS 资源进行建模和设置，从而减少创建和管理资源和基础架构所花费的时间。您可以创建一个描述所需所有 AWS 资源的模板，并为您预 CloudFormation 置和配置这些资源。

使用时 CloudFormation，您可以重复使用模板来一致且重复地设置 Application Auto Scaling 资源。只需描述一次您的资源，然后在多个 AWS 账户 区域中一遍又一遍地配置相同的资源。

Application Auto Scaling 和 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 CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述了您要在 CloudFormation 堆栈中配置的资源。如果你不熟悉 JSON 或 YAML，可以使用 D CloudFormation esigner 来帮助你开始使用 CloudFormation 模板。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的什么是 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**）。要获取服务命名空间，请参阅AWS::ApplicationAutoScaling::ScalableTarget参考资料。
- 与目标资源关联的可扩展维度（例如 **appstream:fleet:DesiredCapacity**）。请参阅AWS::ApplicationAutoScaling::ScalableTarget参考资料以获取可缩放的维度。
- 目标资源的资源 ID（例如 **fleet/sample-fleet**）。有关特定资源的语法和示例的信息，请参阅AWS::ApplicationAutoScaling::ScalableTarget参考资料 IDs。
- 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**）。参见服务相关角色 ARN 参考表格以获取角色 ARNs。

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

示例模板代码段

您可以在《AWS CloudFormation 用户指南》的以下章节中找到要包含在 CloudFormation 模板中的示例片段：

- 有关扩展策略和计划操作的示例，请参阅使用配置应用程序 Auto Scaling 资源 AWS CloudFormation。
- 有关扩展策略的更多示例，请参阅AWS::ApplicationAutoScaling::ScalingPolicy。

了解更多关于 CloudFormation

要了解更多信息 CloudFormation，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- CloudFormation API 引用
- 《AWS CloudFormation 命令行界面用户指南》

Application Auto Scaling 的计划扩缩

通过计划扩缩，您可以根据可预测的负载变化，通过创建在特定时间增加或减少容量的计划操作，为应用程序设置自动扩缩。这使您可以主动扩缩应用程序，以适应可预测的负载变化。

例如，假设您每周遇到规律的流量模式，即负载在一周的中间增加，而在接近周末时会下降。您可以在 Application Auto Scaling 中配置与此模式一致的以下扩缩计划：

- 周三上午，一项计划操作通过增加先前设置的可扩展目标的最小容量来增加容量。
- 周五晚上，另一项计划操作通过降低先前设置的可扩展目标的最大容量来减少容量。

利用这些计划的扩缩操作，您可以优化成本和性能。您的应用程序有足够的容量来处理一周中间的流量高峰，但不会在其他时间过度配置不需要的容量。

您可以同时使用计划的扩缩和扩缩策略，以获得主动和被动扩缩方法的优势。运行计划的扩缩操作后，扩缩策略可以继续决定是否进一步扩缩容量。这有助于确保您有足够的容量来处理应用程序的负载。当您的应用程序扩展以满足需求时，当前容量必须在计划操作设置的最小容量和最大容量范围内。

内容

Application Auto Scaling 计划扩展的工作原理

本主题描述了计划扩展的工作原理，并介绍了要有效使用它所需了解的关键考虑事项。

内容

- 相关资源
- 限制

工作原理

要使用计划扩缩，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。您可以创建仅缩放一次或按循环计划缩放的计划操作。

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。

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 Application Auto Scaling 安排重复性扩展操作。。
- 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 扩缩。
- 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注册 AWS 或自定义资源作为可扩展目标（Application Auto Scaling 可以扩展的资源），以及暂停和恢复扩展。
- put-scheduled-action添加或修改现有可扩展目标的计划操作。
- describe-scaling-activities返回有关某个 AWS 区域中扩展活动的信息。
- describe-scheduled-actions返回有关某个 AWS 区域中计划操作的信息。
- delete-scheduled-action删除计划操作。

相关资源

有关使用定时扩展的详细示例，请参阅 C AWS ompute Blog 上的博客文章 “计划 AWS Lambda 预配置并发以了解反复出现的峰值使用量”。

有关为自动扩缩组创建计划操作的信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 Amazon EC2 Auto Scaling 的计划扩缩。

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。
- 在可扩展资源上查看、添加、更新或移除计划操作的控制台访问权限取决于您使用的资源。有关更多信息，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用。

使用 Application Auto Scaling 创建计划操作 AWS CLI

以下示例说明如何使用 AWS CLI put-scheduled-action命令创建计划操作。当您指定新容量时，可指定最小容量和/或最大容量。

这些示例对与 Application Auto Scaling 集成的一些服务使用可扩展目标。要使用不同的可扩展目标，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。

使用时 AWS CLI，请记住您的命令在 AWS 区域 配置文件中运行。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。

示例

- 创建仅发生一次的计划操作
- 创建按重复间隔运行的计划操作
- 创建按重复计划运行的计划操作
- 创建指定时区的一次性计划操作
- 创建指定时区的重复计划操作

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
```

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 EventBridge 用户指南中的费率表达式。

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/my-cluster/my-service \
  --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/my-cluster/my-service ^
  --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 Application Auto Scaling 安排重复性扩展操作。。

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，其中提供了有关呼叫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 Application Auto Scaling 安排重复性扩展操作。。

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
```

使用 Application Auto Scaling 的定时扩展 AWS CLI

这些示例 AWS CLI 命令描述了使用与 Application Auto Scaling 集成的服务中的资源进行的扩展活动和计划操作。对于不同的可扩展目标，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。

使用时 AWS CLI，请记住您的命令在 AWS 区域 配置文件中运行。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。

示例

- 描述服务的扩展活动
- 描述服务的计划操作
- 描述可扩展目标的计划操作

描述服务的扩展活动

要查看指定服务命名空间中所有可扩展目标的扩展活动，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
```

Output

如果该命令成功，则将返回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。

描述服务的计划操作

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
```

Output

如果该命令成功，则将返回类似于以下内容的输出。

```
{
    "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该`--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
```

Output

如果该命令成功，则将返回类似于以下内容的输出。如果为 `--scheduled-action-names` 提供了多个值，则输出包括其名称匹配的所有计划操作。

```
{
    "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
        }
    ]
}
```

使用 Application Auto Scaling 安排重复性扩展操作。

> **Important**
>
> 如需 Amazon EC2 Auto Scaling 的 cron 表达式的帮助，请参阅 Amazon EC2 Auto Scaling 用户指南中的定期计划主题。使用 Amazon EC2 Auto Scaling，您可以使用传统的 cron 语法，而不是 Application Auto Scaling 使用的自定义 cron 语法。

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。有关使用创建计划操作 AWS CLI 的命令示例，请参阅创建指定时区的重复计划操作。

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 ? * MON *` 会配置一个将于每周一早上 6:30 重复执行的计划操作。星号用作通配符，以匹配字段的所有值。

有关 Application Auto Scaling 计划操作的 cron 语法的更多信息，请参阅亚马逊 EventBridge 用户指南中的 Cron 表达式参考。

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。
- 使用或创建 AWS CLI 或更新计划操作时，必须以 UTC 格式设置开始时间和结束时间。 AWS SDKs

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	意义
0	10	*	*	?	*	每天上午的 10:00（UTC）运行
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。

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 扩缩。

暂停计划扩展

使用带`--suspended-state`选项的register-scalable-target命令并指定`true`为`ScheduledScalingSuspended`属性的值，在可扩展目标上暂停定时缩放，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}"
```

Output

如果该命令成功，则将返回可扩展目标的 ARN。下面是示例输出。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

恢复计划扩展

要恢复计划扩展，请再次运行 register-scalable-target 命令，同时指定 `false` 作为 `ScheduledScalingSuspended` 属性的值。

使用 Application Auto Scaling 的计划操作删除 AWS CLI

完成计划的操作后，可以将其删除。

删除计划操作

使用 delete-scheduled-action 命令。如果成功，此命令不返回任何输出。

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
```

撤消可扩展目标的注册

如果您也完成了可扩展目标，则可以将其取消注册。使用以下 deregister-scalable-target 命令。如果有任何扩展策略或计划操作尚未删除，则可通过此命令删除它们。如果成功，此命令不返回任何输出。

Linux、macOS 或 Unix

```
aws application-autoscaling deregister-scalable-target \
  --service-namespace ec2 \
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity \
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE
```

Windows

```
aws application-autoscaling deregister-scalable-target ^
  --service-namespace ec2 ^
  --scalable-dimension ec2:spot-fleet-request:TargetCapacity ^
  --resource-id spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-37294EXAMPLE
```

Application Auto Scaling 的目标跟踪扩缩策略

目标跟踪扩缩策略根据目标指标值扩缩您的应用程序。这使您的应用程序无需人工干预即可保持最佳性能和成本效益。

通过目标跟踪，您可以选择一个指标和一个目标值，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。Application Auto Scaling 创建并管理在指标偏离目标时触发扩展事件的 CloudWatch警报。这与恒温器保持目标温度的方式类似。

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，当 CPU 使用率超过 50% 时，Application Auto Scaling 将横向扩展（增加容量），以处理增加的负载。当 CPU 利用率降至 50% 以下时，Application Auto Scaling 将横向缩减（减少容量），以便在利用率低的时期优化成本。

目标跟踪策略无需手动定义 CloudWatch 警报和缩放调整。Application Auto Scaling 会根据您设定的目标自动处理这个问题。

您可以使用预定义的指标或自定义指标，设定目标跟踪策略：

- 预定义指标：Application Auto Scaling 提供的指标，例如平均 CPU 利用率或每个目标的平均请求数。
- 自定义指标-您可以使用指标数学来组合指标、利用现有指标或使用自己发布到 CloudWatch的自定义指标。

选择一个与可扩展目标容量的变化成反比的指标。因此，如果您将容量增加一倍，则指标值将减少 50%。这使指标数据能够准确触发按比例扩缩事件。

内容

Application Auto Scaling 目标跟踪扩展的工作原理

本主题描述了目标跟踪扩展的工作原理，并介绍了目标跟踪扩展策略的关键要素。

内容

工作原理

要使用目标跟踪扩展，请创建目标跟踪扩展策略并指定以下内容：

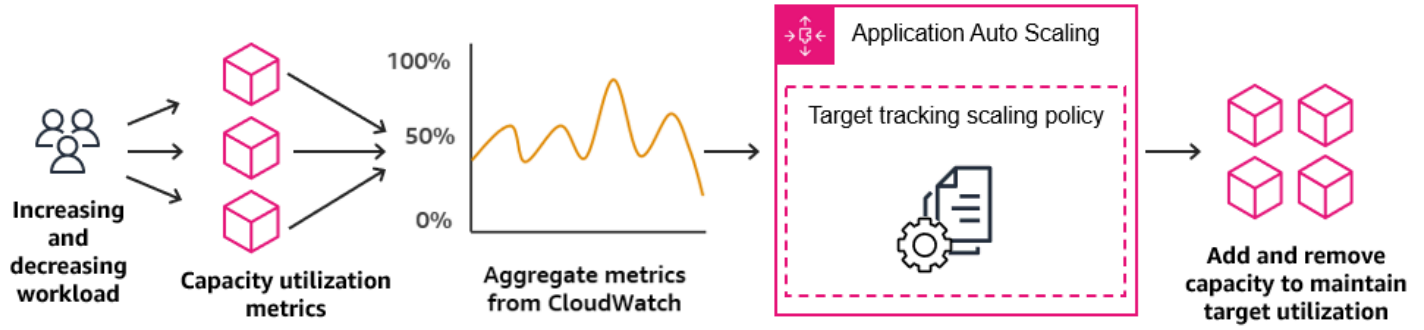
- 指标-要跟踪的 CloudWatch 指标，例如平均 CPU 利用率或每个目标的平均请求数。
- 目标值：指标的目标值，例如 50% 的 CPU 利用率或每个目标每分钟 1000 个请求。

Application Auto Scaling 创建和管理调用扩展策略的 CloudWatch 警报，并根据指标和目标值计算扩展调整。扩缩策略将根据需要增加或减少容量，将指标保持在指定的目标值或接近指定的目标值。

当指标高于目标值时，Application Auto Scaling 会通过增加容量来缩小指标值和目标值之间的差异，从而横向扩展。当指标低于目标值时，Application Auto Scaling 会通过减少容量来横向缩减。

扩缩活动在两者之间有冷却时间，以防止容量快速波动。您可以选择为扩缩策略配置冷却时间。

下图概述显示设置完成时目标跟踪扩缩策略的工作原理。



请注意，目标跟踪扩缩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，如果策略计算出移除最小容量单位可能会使该指标恢复到目标值以上，则该策略不会缩小规模。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。

选择指标

您可以使用预定义的指标或自定义指标，创建目标跟踪扩展策略。

使用预定义指标类型创建目标跟踪扩展策略时，您可以从 目标跟踪扩展策略的预定义目标 中的预定义指标列表中选择一个指标。

选择指标时请记住原则：

- 并非所有自定义指标都适用于目标跟踪。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。
- 当某个指标将实数 0 值发送到 CloudWatch（例如`ALBRequestCountPerTarget`）时，当您的应用程序持续一段时间内没有流量时，Application Auto Scaling 可以缩减到 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。
- 您可以使用指标数学组合现有指标，而不必发布要在扩缩策略中使用的新指标。有关更多信息，请参阅 使用指标数学为 Application Auto Scaling 创建目标跟踪扩展策略。
- 要查看您使用的服务是否支持在服务控制台中指定自定义指标，请参阅该服务的文档。

- 我们建议您使用每隔一分钟可用的指标，以帮助您更快地扩展以响应利用率变化。目标跟踪将评估所有预定义指标和自定义指标的以一分钟为粒度聚合的指标，但底层指标发布数据的频率可能会降低。例如，默认情况下，所有 Amazon EC2 指标都以五分钟为间隔发送，但可配置为每隔一分钟（即详细监控）发送。是否选择此配置取决于单个服务。大部分情况下可尝试使用尽可能小的间隔。

定义目标值

创建目标跟踪扩缩策略时，必须指定一个目标值。目标值表示应用程序的最佳平均利用率或吞吐量。为了经济高效地使用资源，目标值的设置应尽可能高，并为流量的意外增加提供合的理缓冲。当应用程序针对正常流量进行最佳横向扩展时，实际指标值应等于或略低于目标值。

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

您可以选择在目标跟踪扩展策略中定义冷却时间。

冷却时间指定了扩展策略等待上一个扩展活动生效的时间量。

冷却时间有两种类型：

- 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 使用扩展策略成功横向扩展后，它将开始计算冷却时间。除非触发更大的横向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。尽管此横向扩展冷却时间有效，但启动横向扩展活动所添加的容量将计算为下一个横向扩展活动所需容量的一部分。
- 使用横向缩减冷却时间，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，横向缩减活动会被阻止。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，横向缩减冷却时间会停止而不完成。

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。

默认值

Application Auto Scaling 为 ElastiCache 以下可扩展目标提供了默认值 600，为以下可扩展目标提供了默认值 300：

- WorkSpaces 应用程序舰队
- Aurora 数据库集群
- ECS 服务
- Neptune 集群
- SageMaker AI 端点变体
- SageMaker AI 推理组件
- SageMaker AI 无服务器配置的并发性
- Spot Fleets
- Pool of WorkSpaces
- 自定义资源

对于所有其他可扩展目标，默认值为 0 或 null：

- Amazon Comprehend 文档分类和实体识别程序终端节点
- DynamoDB 表和全局二级索引
- Amazon Keyspaces 表
- Lambda 预配置并发
- Amazon MSK 代理存储

Application Auto Scaling 评估冷却时间时，会将 null 值视为零值。

您可以更新任何默认值（包括 null 值），以设置自己的冷却时间。

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展策略一起使用的 CloudWatch 警报。Application Auto Scaling 创建和管理与目标跟踪扩展策略关联的 CloudWatch 警报，并在不再需要时将其删除。
- 如果指标缺少数据点，则会导致 CloudWatch 警报状态更改为`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。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的配置 CloudWatch 警报如何处理丢失的数据。

- 如果设计为很少报告指标，则指标数学可能会很有帮助。例如，要使用最新的值，则使用 `FILL(m1,REPEAT)` 函数，其中 `m1` 是指标。
- 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。

  例如，假设您将 CPU 使用率的目标值设置为 50%，然后自动扩缩组超过了该目标。我们可以确定，添加 1.5 个实例会将 CPU 使用率降低到接近 50%。由于不可能添加 1.5 个实例，我们将该值向上取整，添加两个实例。这可能会将 CPU 使用率降至 50% 以下，但可确保应用程序具有充足的支持资源。类似地，如果我们确定移除 0.5 个实例可将 CPU 利用率提高到 50% 以上，那么在指标降低到我们认为横向缩减不会导致振荡之前，我们不会选择进行横向缩减。

  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横向扩展。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横向扩展。

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横向缩减部分，以便使用与横向扩展不同的方法进行横向缩减。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。

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。

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注册 AWS 或自定义资源作为可扩展目标（Application Auto Scaling 可以扩展的资源），以及暂停和恢复扩展。
- put-scaling-policy为现有可扩展目标添加或修改扩展策略。
- describe-scaling-activities返回有关某个 AWS 区域中扩展活动的信息。
- describe-scaling-policies返回有关某个 AWS 区域中扩展策略的信息。
- delete-scaling-policy删除扩展策略。

相关资源

有关为自动扩缩组创建目标跟踪扩缩策略的信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 Amazon EC2 Auto Scaling 的目标跟踪扩缩策略。

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 AWS Auto Scaling 扩展计划。
- 在可扩展资源上查看、添加、更新或移除目标跟踪扩缩策略的控制台访问权限取决于您使用的资源。有关更多信息，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用。

使用 Application Auto Scaling 创建目标跟踪扩展策略 AWS CLI

此示例使用 AWS CLI 命令为 Amazon EC2 竞价型队列创建目标机架策略。对于不同的可扩展目标，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。

使用时 AWS CLI，请记住您的命令在 AWS 区域 配置文件中运行。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。

任务

步骤 1：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。将每个 *user input placeholder* 替换为您自己的信息。

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
```

Output

如果成功，该命令会返回可扩展目标的 ARN。下面是示例输出。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

步骤 2：创建目标跟踪扩展策略

要创建目标跟踪扩展策略，可以使用以下示例来协助您开始。

创建目标跟踪扩展策略

1. 使用以下 cat 命令可以在您主目录的名为 config.json 的 JSON 文件中存储扩展策略的目标值和预定义指标规范。以下是将 CPU 平均使用率保持在 50% 的示例目标跟踪配置。

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
    }
}
```

有关更多信息，请参阅《App licati PredefinedMetricSpecificationon Auto Scaling API 参考》中的。

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展。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 100。

```
$ cat ~/config.json
{
   "TargetValue": 100.0,
   "CustomizedMetricSpecification":{
      "MetricName": "MyUtilizationMetric",
      "Namespace": "MyNamespace",
      "Dimensions": [
         {
            "Name": "MyOptionalMetricDimensionName",
            "Value": "MyOptionalMetricDimensionValue"
         }
      ],
      "Statistic": "Average",
      "Unit": "Percent"
   }
}
```

有关更多信息，请参阅《App licati CustomizedMetricSpecificationon Auto Scaling API 参考》中的。

2. 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 cpu50-target-tracking-scaling-policy 的扩展策略。

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 cpu50-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 cpu50-target-tracking-scaling-policy --policy-type
 TargetTrackingScaling ^
  --target-tracking-scaling-policy-configuration file://config.json
```

Output

如果成功，此命令将返回代表您创建的两个 CloudWatch 警报的 ARNs 和名称。下面是示例输出。

```
{
    "PolicyARN": "arn:aws:autoscaling:region:account-
id:scalingPolicy:policy-id:resource/ec2/spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE:policyName/cpu50-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"
```

```
        }
    ]
}
```

步骤 3：描述目标跟踪扩展策略

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展策略。

```
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 AWS Command Line Interface 用户指南中的控制 AWS 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`]"
```

Output

下面是示例输出。

```
[
    {
        "PolicyARN": "PolicyARN",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
            },
            "TargetValue": 50.0
        },
        "PolicyName": "cpu50-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
    }
]
```

使用 Application Auto Scaling 的目标跟踪扩展策略 AWS CLI

在使用完目标跟踪扩展策略后，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。它还会删除 Application Auto Scaling 代表你创建的 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 cpu50-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 cpu50-target-tracking-scaling-policy
```

使用指标数学为 Application Auto Scaling 创建目标跟踪扩展策略

使用指标数学，您可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的时间序列。您可以在 CloudWatch 控制台中直观显示生成的时间序列，并将其添加到控制面板中。有关指标数学的更多信息，请参阅 Amazon CloudWatch 用户指南中的使用指标数学。

以下考虑因素适用于指标数学表达式：

- 您可以查询任何可用的 CloudWatch 指标。每个指标都是指标名称、命名空间和零个或多个维度的唯一组合。
- 您可以使用任何算术运算符 (+-*/^)、统计函数（例如 AVG 或 SUM）或其他支持的函数。CloudWatch
- 您可以在数学表达式的公式中同时使用指标和其他数学表达式的结果。
- 指标规范中使用的任何表达式最终都必须返回一个单个时间序列。
- 您可以使用 CloudWatch 控制台或 CloudWatch GetMetricDataAPI 验证指标数学表达式是否有效。

主题

示例：每个任务的 Amazon SQS 队列积压

要计算每个任务的 Amazon SQS 队列积压，请获取可用于从队列中检索的消息的大致数量，然后将该数字除以服务中运行的 Amazon ECS 任务的数量。有关更多信息，请参阅计算博客上使用自定义指标的亚马逊弹性容器服务 (ECS) A AWS uto Sc aling。

表达式的逻辑如下：

```
sum of (number of messages in the queue)/(number of tasks that are
currently in the RUNNING state)
```

那么您的 CloudWatch 指标信息如下所示。

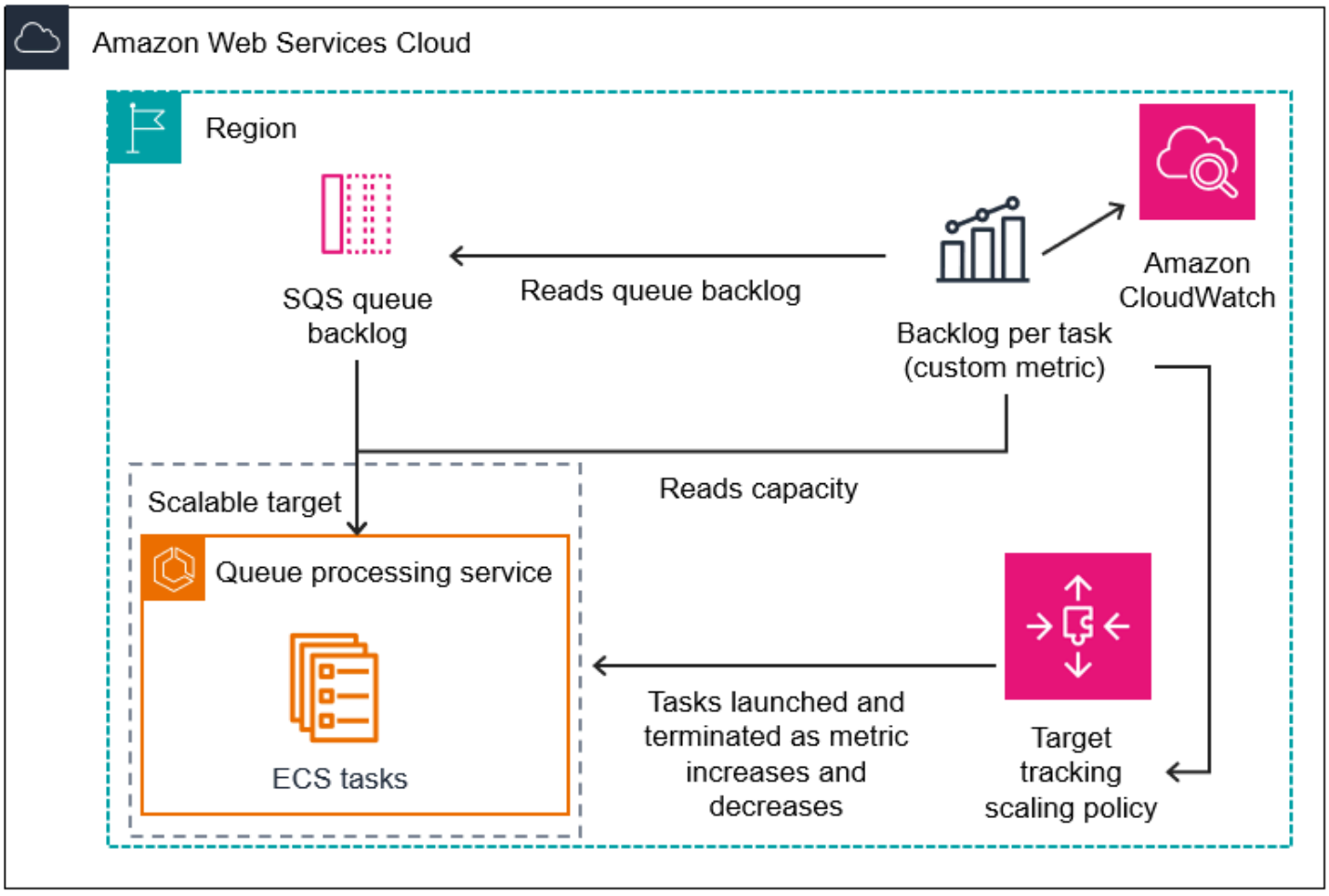
ID	CloudWatch 指标	Statistic	周期
m1	ApproximateNumberOfMessagesVisible	Sum	1 minute

ID	CloudWatch 指标	Statistic	周期
m2	RunningTaskCount	平均值	1 minute

您的指标数学 ID 和表达式如下所示。

ID	Expression
e1	`(m1)/(m2)`

下图阐明了此指标的架构：



使用该指标数学来创建目标跟踪扩展策略 (AWS CLI)

1. 将指标数学表达式作为自定义指标规范的一部分存储在名为 `config.json` 的 JSON 文件中。

使用下面的示例帮助您快速开始。将每个 *user input placeholder* 替换为您自己的信息。

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Get the queue size (the number of messages waiting to be
 processed)",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "ApproximateNumberOfMessagesVisible",
                        "Namespace": "AWS/SQS",
                        "Dimensions": [
                            {
                                "Name": "QueueName",
                                "Value": "my-queue"
                            }
                        ]
                    },
                    "Stat": "Sum"
                },
                "ReturnData": false
            },
            {
                "Label": "Get the ECS running task count (the number of currently
 running tasks)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "RunningTaskCount",
                        "Namespace": "ECS/ContainerInsights",
                        "Dimensions": [
                            {
                                "Name": "ClusterName",
                                "Value": "my-cluster"
                            },
                            {
                                "Name": "ServiceName",
                                "Value": "my-service"
                            }
                        ]
                    },
```

```
                    "Stat": "Average"
                },
                "ReturnData": false
            },
            {
                "Label": "Calculate the backlog per instance",
                "Id": "e1",
                "Expression": "m1 / m2",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
}
```

有关更多信息，请参阅《App licati TargetTrackingScalingPolicyConfigurationon Auto Scaling API 参考》中的。

> **Note**
>
> 以下是一些其他资源，可以帮助您查找指标名称、命名空间、维度和指标 CloudWatch 统计信息：
>
> - 有关 AWS 服务的可用指标的信息，请参阅《亚马逊 CloudWatch 用户指南》中发布 CloudWatch 指标的AWS 服务。
> - 要使用获取指标的确切指标名称、命名空间和维度（如果适用）AWS CLI，请参阅列表 CloudWatch 指标。

2. 要创建此策略，请使用 JSON 文件作为输入运行put-scaling-policy命令，如以下示例所示。

```
aws application-autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-scaling-policy \
  --service-namespace ecs --scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service \
  --policy-type TargetTrackingScaling --target-tracking-scaling-policy-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon 资源名称 (ARN) 和代表您创建 ARNs 的两个 CloudWatch 警报中的一个。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/my-cluster/my-
service:policyName/sqs-backlog-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/my-cluster/my-service-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/my-cluster/my-service-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/my-cluster/my-service-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/my-cluster/my-service-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

> **Note**
>
> 如果此命令引发错误，请确保已将 AWS CLI 本地版本更新到最新版本。

限制

- 最大请求大小为 50KB。这是您在策略定义中使用公制数学时 PutScalingPolicyAPI 请求的总有效负载大小。如果您超过此限制，Application Auto Scaling 会拒绝该请求。
- 结合使用指标数学与目标跟踪扩缩策略时，不支持以下服务：
  - Amazon Keyspaces（Apache Cassandra 兼容）
  - DynamoDB
  - Amazon EMR
  - Amazon MSK
  - Amazon Neptune

Application Auto Scaling 的分步扩缩策略

分步扩展策略根据 CloudWatch 警报以预定义的增量扩展应用程序的容量。您可以定义单独的扩缩策略，以便在超过警报阈值时处理横向扩展（增加容量）和横向缩减（减少容量）。

使用分步扩展策略，您可以创建和管理调用扩展过程的 CloudWatch 警报。当警报被触发时，Application Auto Scaling 会启动与该警报关联的扩缩策略。

分步扩缩策略使用一组调整（称为分步调整）来扩缩容量。调整的大小将根据超出警报阈值的规模而变化。

- 如果违例超过第一个阈值，Application Auto Scaling 将应用第一步调整。
- 如果违例超过第二个阈值，Application Auto Scaling 将应用第二步调整，以此类推。

这使扩缩策略能够针对警报指标的微小和重大变化作出适当响应。

当扩缩活动正在进行中时，该策略将继续响应其他警报。这意味着 Application Auto Scaling 将在所有警报发生时对其进行评估。冷却时间用于防止由于快速连续发生多个警报而导致的过度扩缩。

与目标跟踪一样，分步扩缩可以帮助在流量发生变化时自动扩缩应用程序的容量。但是，目标跟踪策略往往更易于实施和管理，以满足稳定的扩缩需求。

支持的可扩展目标

您可以将分步扩缩策略与以下可扩展目标配合使用：

- WorkSpaces 应用程序舰队
- Aurora 数据库集群
- ECS 服务
- EMR 集群
- SageMaker AI 端点变体
- SageMaker AI 推理组件
- SageMaker AI 无服务器配置的并发性
- Spot Fleets
- 自定义资源

内容

Application Auto Scaling 步进扩展策略的工作原理

本主题描述步进扩展的工作原理，并介绍步进扩展策略的关键元素。

内容

工作原理

要使用步进缩放，您需要创建一个 CloudWatch 警报，用于监控可扩展目标的指标。定义确定触发警报的指标、阈值和评估周期数。您还可以创建分步扩缩策略，在其中定义在突破警报阈值时如何扩缩容量并将其与可扩展目标相关联。

在策略中添加分步调整。您可以根据警报的阈值突破大小定义不同的分步调整。例如：

- 如果警报指标达到 60%，则横向扩展 10 个容量单位
- 如果警报指标达到 75%，则横向扩展 30 个容量单位
- 如果警报指标达到 85%，则横向扩展 40 个容量单位

当在指定的评估周期数内超过警报阈值时，Application Auto Scaling 将应用策略中定义的分步调整。针对其他警报触发情况，可以继续进行调整，直到警报状态恢复为 OK。

扩缩活动在两者之间有冷却时间，以防止容量快速波动。您可以选择为扩缩策略配置冷却时间。

分步调整

在创建分步扩缩策略时，您可以指定一个或多个分步调整，它们会动态地根据警报违规的大小自动扩缩目标容量。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 CloudWatch 警报关联的指标的统计数据聚合指标数据点。超过警报时，将调用相应的扩缩策略。Application Auto Scaling 将您指定的聚合类型应用于来自的最新指标数据点 CloudWatch （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，假设您针对指标高于 50% 时发出了 CloudWatch 警报并制定了扩展策略。然后，当指标低于 50% 时，又发出了另一个警报并采取横向缩减策略。您进行了一组分步调整，每个策略的调整类型为 `PercentChangeInCapacity`：

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
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%
-------------------------------------------------------------------------
```

现在，假设您有一个容量为 10 的可扩展目标，使用基于此目标的扩缩配置。以下几点总结了扩展配置相对于可扩展目标的容量的行为：

- 在聚合指标值大于 40 且小于 60 时，将保留原始容量。
- 如果指标值达到 60，则 Application Auto Scaling 将可扩展目标的容量增加 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分步调整（减去 14 的 10%，即 1.4，向下舍入到 1），Application Auto Scaling 将可扩展目标的容量减少 1，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)。如果一个分步调整有正上限，则必须有一个分步调整有空上限。
- 同一分步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—按指定值增加或减少可扩展目标的当前容量。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值。为此调整类型指定一个非负值。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—按指定的百分比增加或减少可扩展目标的当前容量。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）。

  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。

  使用 PercentChangeInCapacity，您还可以使用`MinAdjustmentMagnitude`参数指定最小缩放量。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

您可以选择在分步扩展策略中定义冷却时间。

冷却时间指定了扩展策略等待上一个扩展活动生效的时间量。

有两种方法可以计划分步扩展配置的冷却时间使用：

- 使用横向扩展策略的冷却时间，目的是持续（但不过度）横向扩展。Application Auto Scaling 使用扩展策略成功横向扩展后，它将开始计算冷却时间。除非触发更大的横向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。尽管此横向扩展冷却时间有效，但启动横向扩展活动所添加的容量将计算为下一个横向扩展活动所需容量的一部分。
- 使用横向缩减策略冷却时间，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，因此在横向缩减冷却时间过期之前，横向缩减活动会被阻止。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，横向缩减冷却时间会停止而不完成。

例如，出现流量高峰时会触发警报，Application Auto Scaling 会自动增加容量以帮助处理增加的负载。如果为横向扩展策略设置冷却时间，当警报触发策略以将容量增加 2 时，扩展活动成功完成，横向扩展冷却时间开始。如果在冷却时间内再次触发警报，但进行了 3 这样的更大幅度的步进调整，之前增加的 2 将视为当前容量的一部分。因此，仅在容量中增加 1。与等待冷却时间过期相比，这可以实现更快的扩展，但不会增加超出您需求的容量。

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。

如果未指定值，则默认值为 300。

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注册 AWS 或自定义资源作为可扩展目标（Application Auto Scaling 可以扩展的资源），以及暂停和恢复扩展。
- put-scaling-policy为现有可扩展目标添加或修改扩展策略。
- describe-scaling-activities返回有关某个 AWS 区域中扩展活动的信息。
- describe-scaling-policies返回有关某个 AWS 区域中扩展策略的信息。
- delete-scaling-policy删除扩展策略。

注意事项

使用分步扩缩策略时，需要注意以下事项：

- 考虑是否可以足够准确地预测应用程序上的分步调整，以便使用分步扩缩。如果您的扩缩指标的升高或降低与可扩展目标的容量成比例，则建议您使用目标跟踪扩缩策略。您仍然可以选择使用步进扩展作为附加策略来实现更高级的配置。例如，您可以在利用率达到特定级别时配置更积极的响应。
- 确保在横向扩展和横向缩减之间选择足够的余量，以防止摇摆。摆动是横向缩减和横向扩展的无限循环。也就是说，如果采取扩展操作，则指标值将更改并启动另一个相反方向的扩展操作。

相关资源

有关为自动扩缩组创建分步扩缩策略的信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的 Amazon EC2 Auto Scaling 的步进和简单扩展策略。

控制台访问

在可扩展资源上查看、添加、更新或移除分步扩缩策略的控制台访问权限取决于您使用的资源。有关更多信息，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用。

使用 Application Auto Scaling 创建步进缩放策略 AWS CLI

此示例使用 AWS CLI 命令为 Amazon ECS 服务创建分步扩展策略。对于不同的可扩展目标，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。

使用时 AWS CLI，请记住您的命令在 AWS 区域 配置文件中运行。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。

任务

- 步骤 1：注册可扩展目标
- 步骤 2：创建步进扩展策略
- 步骤 3：创建调用扩展策略的警报

步骤 1：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。将每个 *user input placeholder* 替换为您自己的信息。

Linux、macOS 或 Unix

```
aws application-autoscaling register-scalable-target --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-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/my-cluster/my-service ^
  --min-capacity 2 --max-capacity 10
```

Output

如果成功，该命令会返回可扩展目标的 ARN。下面是示例输出。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

步骤 2：创建步进扩展策略

要为可扩展目标创建步进扩展策略，可以使用以下示例来协助您入门。

Scale out

为横向扩展创建步进扩展策略（增加容量）

1. 使用以下 cat 命令在主目录的名为 config.json 的 JSON 文件中存储步进扩展策略配置。以下是一个配置示例，其调整类型为PercentChangeInCapacity，该配置根据以下步骤调整（假设 CloudWatch 警报阈值为 70）来增加可扩展目标的容量：

   - 当指标的值大于或等于 70 但小于 85 时，将容量增加 10%
   - 当指标的值大于或等于 85 但小于 95 时，将容量增加 20%
   - 当指标的值大于或等于 95 时，将容量增加 30%

```
$ cat ~/config.json
{
  "AdjustmentType": "PercentChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "MinAdjustmentMagnitude": 1,
  "StepAdjustments": [
    {
      "MetricIntervalLowerBound": 0.0,
      "MetricIntervalUpperBound": 15.0,
```

```
      "ScalingAdjustment": 10
    },
    {
      "MetricIntervalLowerBound": 15.0,
      "MetricIntervalUpperBound": 25.0,
      "ScalingAdjustment": 20
    },
    {
      "MetricIntervalLowerBound": 25.0,
      "ScalingAdjustment": 30
    }
  ]
}
```

有关更多信息，请参阅《App licati StepScalingPolicyConfigurationon Auto Scaling API 参考》中的。

2. 使用以下put-scaling-policy命令以及您创建config.json的文件来创建名为的扩展策略my-step-scaling-policy。

   Linux、macOS 或 Unix

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

   Output

   输出包括作为策略唯一名称的 ARN。你需要它来为你的策略创建 CloudWatch 警报。下面是示例输出。

```
{
```

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

Scale in

为横向缩减创建步进扩展策略（减少容量）

1. 使用以下 cat 命令在主目录的名为 config.json 的 JSON 文件中存储步进扩展策略配置。以下是一个配置示例，其调整类型为ChangeInCapacity，根据以下步骤调整（假设 CloudWatch 警报阈值为 50），该配置会降低可扩展目标的容量：

   - 当指标的值小于或等于 50 但大于 40 时，将容量减少 1
   - 当指标的值小于或等于 40 但大于 30 时，将容量减少 2
   - 当指标的值小于或等于 30 时，将容量减少 3

```
$ cat ~/config.json
{
  "AdjustmentType": "ChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "StepAdjustments": [
    {
      "MetricIntervalUpperBound": 0.0,
      "MetricIntervalLowerBound": -10.0,
      "ScalingAdjustment": -1
    },
    {
      "MetricIntervalUpperBound": -10.0,
      "MetricIntervalLowerBound": -20.0,
      "ScalingAdjustment": -2
    },
    {
      "MetricIntervalUpperBound": -20.0,
      "ScalingAdjustment": -3
    }
  ]
}
```

有关更多信息，请参阅《App licati StepScalingPolicyConfigurationon Auto Scaling API 参考》中的。

2. 使用以下put-scaling-policy命令以及您创建config.json的文件来创建名为的扩展策略my-step-scaling-policy。

   Linux、macOS 或 Unix

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

   Output

   输出包括作为策略唯一名称的 ARN。您需要此 ARN 来为您的策略创建 CloudWatch 警报。下面是示例输出。

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

步骤 3：创建调用扩展策略的警报

最后，使用以下 CloudWatch put-metric-alarm命令创建警报，以便与步进缩放策略一起使用。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，请在中指定其名称，在中`--metric-name`指定其命名空间`--namespace`。

Linux、macOS 或 Unix

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-ECS:service/my-cluster/my-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/my-cluster/my-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
```

使用 Application Auto Scaling 描述步进缩放策略 AWS CLI

您可以使用describe-scaling-policies命令描述服务命名空间的所有扩展策略。以下示例描述了所有 Amazon ECS 服务的所有扩展策略。要针对特定 Amazon ECS 服务列出它们，只需添加 `--resource-id` 选项即可。

```
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 AWS Command Line Interface 用户指南中的控制 AWS 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`]"
```

Output

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/my-cluster/my-service",
        "ServiceNamespace": "ecs",
        "Alarms": [
            {
                "AlarmName": "Step-Scaling-AlarmHigh-ECS:service/my-cluster/my-
service",
                "AlarmARN": "arn:aws:cloudwatch:region:012345678910:alarm:Step-Scaling-
AlarmHigh-ECS:service/my-cluster/my-service"
            }
```

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

使用 Application Auto Scaling 的分步扩展策略删除 AWS CLI

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

删除您的扩展策略

使用 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/my-cluster/my-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/my-cluster/my-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/my-cluster/my-service 和 Step-Scaling-AlarmLow-ECS:service/my-cluster/my-service 警报：

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-ECS:service/my-cluster/my-service Step-Scaling-AlarmLow-ECS:service/my-cluster/my-service
```

Application Auto Scaling 的预测扩展

预测性扩展可以主动扩展您的应用程序。预测性扩展可分析历史负载数据，以检测流量中的每日或每周模式。它使用这些信息来预测未来的容量需求，以主动增加应用程序的容量以匹配预期的负载。

预测式扩展非常适合以下情况：

- 周期性流量，例如正常营业时间内的高资源利用率以及晚上和周末的低资源利用率
- 重复 on-and-off的工作负载模式，例如批处理、测试或定期数据分析。
- 初始化需要很长时间的应用程序，从而在向外扩展事件期间对应用程序性能造成明显的延迟影响

内容

Application Auto Scaling 预测缩放的工作原理

要使用预测扩展，请创建预测性扩展策略，指定要监控和分析的 CloudWatch 指标。您可以使用预定义的指标或自定义的指标。要使预测性扩展开始预测未来值，此指标必须包含至少 24 小时的数据。

创建策略后，预测性扩展将开始分析最多过去 14 天的指标数据，以确定模式。其使用此分析生成未来 48 小时容量需求的每小时预测。使用最新 CloudWatch 数据，预测每 6 小时更新一次。随着新数据的出现，预测性扩展能够不断提高未来预测的准确性。

您可以先在 “仅限预测” 模式下启用预测缩放。在此模式下，它会生成容量预测，但实际上不会根据这些预测扩展容量。这使您可以评估预测的准确性和适用性。

查看预测数据并决定根据该数据开始扩展后，将扩展策略切换到预测和扩展模式。在此模式中：

- 如果预测预计负载会增加，则预测性扩展将增加容量。
- 如果预测预计负载会减少，则预测性扩展不会缩减以减少容量。这样可以确保只有在需求实际下降时才缩减规模，而不仅仅是根据预测。要移除不再需要的容量，您必须创建目标跟踪或步进扩展策略，因为它们会响应实时指标数据。

默认情况下，预测性扩展会根据该小时的预测在每小时开始时缩放您的可扩展目标。您可以选择在 PutScalingPolicy API 操作中使用SchedulingBufferTime属性来指定更早的开始时间。这使您能够在预测的需求之前启动预测的容量，从而使新容量有足够的时间准备好处理流量。

最大容量限制

默认情况下，设置扩缩策略后，这些策略无法将容量增加到高于最大容量。

或者，如果预测容量接近或超过可扩展目标的最大容量，则可以允许自动增加可扩展目标的最大容量。要启用此行为，请使用 PutScalingPolicy API 操作中的 MaxCapacityBreachBehavior 和 MaxCapacityBuffer 属性或 AWS 管理控制台中的最大容量行为设置。

> **Warning**
>
> 允许自动增加最大容量时应谨慎行事。最大容量不会自动降回到原来的最大值。

扩缩策略创建、管理和删除的常用命令

使用预测性扩展策略的常用命令包括：

- register-scalable-target将资源注册 AWS 或自定义为可扩展目标、暂停扩展和恢复扩展。
- put-scaling-policy以创建预测性扩展策略。
- get-predictive-scaling-forecast检索预测性扩展策略的预测数据。
- describe-scaling-activities返回有关中扩展活动的信息 AWS 区域。
- describe-scaling-policies以返回有关扩展策略的信息 AWS 区域。
- delete-scaling-policy删除扩展策略。

自定义指标

自定义指标可用于预测应用程序所需的容量。当预定义的指标不足以捕获应用程序的负载时，自定义指标很有用。

注意事项

使用预测性缩放时，需要考虑以下注意事项。

- 确认预测性扩展是否适合您的应用程序。如果应用程序表现出特定于一周中的某一天或一天中的时间的重复负载模式，则该应用程序非常适合预测性扩展。在让预测性扩展主动扩展您的应用程序之前，先评估预测。
- 预测式扩展至少需要 24 小时的历史数据才能开始预测。但是，如果历史数据跨越整整两周，预测会更有效。
- 选择一个负载指标，该指标应准确代表应用程序的全部负载，并且是应用程序中对扩缩最重要的方面。

为 Application Auto Scaling 创建预测性扩展策略

以下示例策略使用 AWS CLI 为 Amazon ECS 服务配置预测性扩展策略。将每个 *user input placeholder* 替换为您自己的信息。

有关您可以指定的 CloudWatch 指标的更多信息，请参阅PredictiveScalingMetricSpecification《Amazon EC2 Auto Scaling API 参考》。

以下是具有预定义内存配置的策略的示例。

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceMemoryUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

以下示例说明了通过使用指定配置文件运行put-scaling-policy命令来创建策略。

```
aws aas put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
```

```
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

如果成功，该命令会返回策略的 ARN。

```
{
"PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
"Alarms": []
}
```

使用计划操作覆盖预测值

有时，您可能会获得有关未来应用程序需求的其他信息，预测计算无法考虑这些信息。例如，预测计算可能会低估即将举行的市场营销活动所需的容量。您可以使用计划操作在未来时段内临时覆盖预测。计划操作可以循环运行，也可以在出现一次性需求波动的特定日期和时间运行。

例如，您可以创建具有高于预测容量的最小容量的计划操作。在运行时，Application Auto Scaling 会更新可扩展目标的最小容量。由于预测式扩展可针对容量进行优化，因此执行最小容量高于预测值的计划操作。这样可以防止容量低于预期。要停止覆盖预测，请使用第二个计划操作将最小容量恢复到其原始设置。

以下过程概述了在将来期间覆盖预测的步骤。

主题

> **Important**
>
> 本主题假设您尝试覆盖预测，以扩展到比预测更高的容量。如果您需要在不受预测性扩展策略干扰的情况下暂时减少容量，则请改用仅预测模式。在仅预测模式下，预测性扩展将继续生成预测，但不会自动增加容量。然后，您可以监控资源利用率，并根据需要手动缩减组大小。

步骤 1：（可选）分析时间序列数据

首先分析预测时间序列数据。这是一个可选步骤，但如果您想了解预测的详细信息，它会很有帮助。

1. 检索预测

   创建预测后，您可以查询预测中的特定时间段。查询的目的是获得特定时间段的时间序列数据的完整视图。

   您的查询最多可以包含两天的未来预测数据。如果您已经使用了一段时间预测式扩展，您还可以访问过去的预测数据。但是，开始和结束时间之间的最长持续时间为 30 天。

   要检索预测，请使用get-predictive-scaling-forecast命令。以下示例获取了 Amazon ECS 服务的预测性扩展预测。

   ```
   aws application-autoscaling get-predictive-scaling-forecast --service-namespace ecs
    \
       --scalable-dimension ecs:service:DesiredCount \
       --resource-id 1234567890abcdef0
       --policy-name predictive-scaling-policy \
       --start-time "2021-05-19T17:00:00Z" \
       --end-time "2021-05-19T23:00:00Z"
   ```

   响应包括两个预测：LoadForecast和CapacityForecast。 LoadForecast显示每小时负荷预测。 CapacityForecast显示按小时计算的容量的预测值，以便在保持指定TargetValue负载的同时处理预测的负载。

2. 确定目标时间段

   确定应发生一次性需求波动的小时数。请记住，预测中显示的日期和时间以 UTC 为单位。

步骤 2：创建两个计划操作

接下来，在应用程序的负载高于预测负载的特定时间段内创建两个计划操作。例如，如果您的营销活动会在有限时间段内使网站的流量增加，则可计划一个一次性操作以在其启动时更新最小容量。然后，安排另一个操作，以便在事件结束时将最小容量返回到原始设置。

为一次性事件创建两个计划操作（AWS CLI）

要创建计划操作，请使用 put-scheduled-action命令。

以下示例定义了 Amazon EC2 Auto Scaling 的计划，该计划在 5 月 19 日下午 5:00 将至少三个实例的容量保持为八小时。以下命令显示如何实现此方案。

第一个put-scheduled-update-group操作命令指示亚马逊 EC2 Auto Scaling 在世界标准时间 2021 年 5 月 19 日下午 5:00 更新指定 Auto Scaling 组的最小容量。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-
start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-
capacity 3
```

第二个命令指示 Amazon EC2 Auto Scaling 将该组的最小容量设置为 2021 年 5 月 20 日 UTC 上午 1:00。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end
 \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-
capacity 1
```

将这些计划操作添加到自动扩缩组后，Amazon EC2 Auto Scaling 将执行以下操作：

- 2021 年 5 月 19 日下午 5:00，第一个计划的操作将运行。如果组中当前已少于三个实例，则该组会扩展到三个实例。在此期间和接下来的八小时内，如果预测容量高于实际容量，或者如果有动态扩展策略生效，Amazon EC2 Auto Scaling 可以继续横向扩展。
- 2021 年 5 月 20 日上午 1:00，将运行第二个计划的操作。这将在事件结束时将最小容量恢复为其原始设置。

根据重复性计划进行扩展

要覆盖每周相同时间段的预测，请创建两个计划操作，并使用 cron 表达式提供时间和日期逻辑。

此 cron 表达式格式包含五个空格分隔的字段：[Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]。字段可以包含任何允许的值，包括特殊字符。

例如，下面的 cron 表达式在每周二上午 6:30 运行操作。星号用作通配符，以匹配字段的所有值。

```
30 6 * * 2
```

使用自定义指标的高级预测性扩展策略

在预测性扩展策略中，您可以使用预定义指标或自定义指标。当预定义的指标不能充分描述您的应用程序负载时，自定义指标会很有用。

使用自定义指标创建预测性扩展策略时，您可以指定由提供的其他 CloudWatch 指标 AWS，也可以指定自己定义和发布的指标。您还可以使用公制数学来汇总现有指标并将其转换为 AWS 不会自动跟踪的新时间序列。例如，通过计算新的总和或平均值来组合数据中的值时，该操作称为执行聚合。生成的数据称为聚合。

以下部分包含了有关如何为构造策略的 JSON 结构的最佳实践和示例。

主题

最佳实践

以下最佳实践可帮助您更有效地使用自定义指标：

- 对于负载指标规范，最有用的指标是表示应用程序负载的指标。
- 扩展指标必须与容量成反比。也就是说，如果可扩展目标增加，则扩展指标的减少比例应大致相同。为确保预测性扩展按预期采取行动，负载指标和扩展指标还必须彼此之间密切关联。
- 目标利用率必须与扩展指标的类型匹配。对于使用 CPU 利用率的策略配置，这是目标百分比。对于使用吞吐量（例如请求数或消息数）的策略配置，这是在任何一分钟间隔内每个实例的目标请求数或目标消息数。
- 如果未遵循这些建议，那么时间序列的预测未来值可能会不正确。要验证数据是否正确，您可以查看预测值。或者，在创建预测性扩展策略后，检查调用 GetPredictiveScalingForecastAPI 返回的`LoadForecast`和`CapacityForecast`对象。
- 我们强烈建议您在仅预测模式下配置预测式扩展，以便在预测式扩展开始主动扩展容量之前对预测进行评估。

先决条件

要将自定义指标添加到预测性扩缩策略，您必须具有 `cloudwatch:GetMetricData` 权限。

要指定您自己的指标而不是 AWS 提供的指标，您必须先将指标发布到 CloudWatch。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的发布自定义指标。

如果发布自己的指标，请确保以至少五分钟的频率发布数据点。Application Auto Scaling CloudWatch 根据所需的周期长度从中检索数据点。例如，负载指标规范使用每小时指标来衡量应用程序的负载。CloudWatch 使用您发布的指标数据，通过将所有数据点与每个一小时内的时间戳聚合在一起，为任何一小时的时间段提供单个数据值。

构造自定义指标的 JSON

以下部分包含有关如何配置预测扩展以查询 Amazon EC2 Auto Scaling 数据的示例。 CloudWatch 配置此选项有两种不同的方法，您选择的方法会影响您为预测性扩缩策略构造 JSON 时使用的格式。使用指标数学时，JSON 格式会根据所执行的指标数学进一步变化。

1. 要创建可直接从提供的其他 CloudWatch 指标 AWS 或您发布到的指标中获取数据的策略 CloudWatch，请参阅包含自定义负载和扩缩指标的预测性扩缩策略示例（AWS CLI）。
2. 要创建可查询多个 CloudWatch 指标并使用数学表达式根据这些指标创建新时间序列的策略，请参阅使用指标数学表达式。

包含自定义负载和扩缩指标的预测性扩缩策略示例（AWS CLI）

要使用创建带有自定义负载和扩展指标的预测性扩展策略 AWS CLI，请将的参数存储`--predictive-scaling-configuration`在名为的 JSON 文件中`config.json`。

您可以将以下示例中的可替换值替换为您的指标和目标利用率值，从而开始添加自定义指标。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
```

```
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Sum"
            }
          }
        ]
      }
    }
  ]
}
```

有关更多信息，请参阅MetricDataQuery《亚马逊 EC2 Auto Scaling API 参考》。

> **Note**
>
> 以下是一些其他资源，可以帮助您查找指标名称、命名空间、维度和指标 CloudWatch 统计信息：

- 有关 AWS 服务的可用指标的信息，请参阅《亚马逊 CloudWatch 用户指南》中发布 CloudWatch 指标的AWS 服务。
- 要使用获取指标的确切指标名称、命名空间和维度（如果适用）AWS CLI，请参阅列表 CloudWatch 指标。

要创建此策略，请使用 JSON 文件作为输入运行put-scaling-policy命令，如以下示例所示。

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令将返回策略的 Amazon 资源名称（ARN）。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
}
```

使用指标数学表达式

以下部分提供了预测性扩缩策略的信息和示例，这些示例演示了如何在策略中使用指标数学。

主题

- 了解指标数学
- Amazon EC2 Auto Scaling 的预测性扩展策略示例，该策略使用公制数学组合指标 (AWS CLI)
- blue/green 部署场景中使用的预测性扩展策略示例 (AWS CLI)

了解指标数学

如果您只想汇总现有指标数据，那么公 CloudWatch 制数学可以为您节省向其发布另一个指标的工作量和成本 CloudWatch。您可以使用 AWS 提供的任何指标，也可以使用在应用程序中定义的指标。

有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用公制数学。

如果您选择在预测性扩展策略中使用指标数学表达式，请考虑以下几点：

- 指标数学运算使用指标名称、命名空间和维度键/值对指标的唯一组合的数据点。

- 您可以使用任何算术运算符 (+-*/^)、统计函数（例如 AVG 或 SUM）或其他支持的函数。CloudWatch
- 您可以在数学表达式的公式中同时使用指标和其他数学表达式的结果。
- 指标数学表达式可以由不同的聚合组成。但是，得到最终聚合结果的最佳实践是针对扩展指标使用 `Average` 以及针对负载指标使用 `Sum`。
- 指标规范中使用的任何表达式最终都必须返回一个单个时间序列。

要使用指标数学，请执行以下操作：

- 选择一个或多个 CloudWatch 指标。然后，创建表达式。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的使用公制数学。
- 使用 CloudWatch控制台或 CloudWatch GetMetricDataAPI 验证指标数学表达式是否有效。

Amazon EC2 Auto Scaling 的预测性扩展策略示例，该策略使用公制数学组合指标 (AWS CLI)

有时，您可能需要首先以某种方式处理其数据，而不是直接指定指标。例如，您可能有一个从 Amazon SQS 队列中提取工作的应用程序，并且可能希望使用队列中的项目数作为预测性扩展的标准。队列中的消息数不仅仅定义您需要的实例数。因此，需要执行更多工作来创建可用于计算每个实例的积压的指标。

以下示例是适用于此场景的预测扩展策略示例。它指定了基于 Amazon SQS `ApproximateNumberOfMessagesVisible` 指标的扩展和负载指标，即可从队列中获取的用于检索的消息数量。它还使用 Amazon EC2 Auto Scaling `GroupInServiceInstances` 指标和数学表达式，计算扩展指标的每个实例的积压。

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
```

```
            "Metric": {
              "MetricName": "ApproximateNumberOfMessagesVisible",
              "Namespace": "AWS/SQS",
              "Dimensions": [
                {
                  "Name": "QueueName",
                  "Value": "my-queue"
                }
              ]
            },
            "Stat": "Sum"
          },
          "ReturnData": false
        },
        {
          "Label": "Get the group size (the number of running instances)",
          "Id": "running_capacity",
          "MetricStat": {
            "Metric": {
              "MetricName": "GroupInServiceInstances",
              "Namespace": "AWS/AutoScaling",
              "Dimensions": [
                {
                  "Name": "AutoScalingGroupName",
                  "Value": "my-asg"
                }
              ]
            },
            "Stat": "Sum"
          },
          "ReturnData": false
        },
        {
          "Label": "Calculate the backlog per instance",
          "Id": "scaling_metric",
          "Expression": "queue_size / running_capacity",
          "ReturnData": true
        }
      ]
    },
    "CustomizedLoadMetricSpecification": {
      "MetricDataQueries": [
        {
          "Id": "load_metric",
```

```
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
              },
              "Stat": "Sum"
            },
            "ReturnData": true
          }
        ]
      }
    }
  ]
}
```

该示例返回策略的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

blue/green 部署场景中使用的预测性扩展策略示例 (AWS CLI)

搜索表达式提供了一个高级选项，您可以在其中查询多个 Auto Scaling 组中的指标并对其执行数学表达式。这对于 blue/green 部署特别有用。

> **Note**
>
> 蓝/绿部署是一种部署方法，您可以在其中创建两个独立但相同的 Auto Scaling 组。只有其中一个组接收生产流量。用户流量最初定向到较早的Auto Scaling 组（“蓝色”），而新组（“绿色”）用于测试和评估应用程序或服务的新版本。测试并接受新部署后，用户流量将转移到“绿色”的 Auto Scaling 组。然后，您可以在部署成功后删除“蓝色”组。

在 blue/green 部署过程中创建新的 Auto Scaling 组时，每个组的指标历史记录可以自动包含在预测性扩展策略中，而无需更改其指标规范。有关更多信息，请参阅 C AWS ompute 博客上的将 EC2 Auto Scaling 预测性扩展策略用于蓝/绿部署。

以下示例策略说明了如何执行此操作。在此示例中，策略使用 Amazon EC2 发出的 CPUUtilization 指标。它使用 Amazon EC2 Auto Scaling GroupInServiceInstances 指标和数学表达式，计算每个实例的扩展指标的值。它还指定了一个容量指标规范来获取 GroupInServiceInstances 指标。

根据指定的搜索条件，搜索表达式查找多个 Auto Scaling 组中实例的 CPUUtilization。如果您稍后创建了匹配相同搜索条件的新 Auto Scaling 组，则自动包含新 Auto Scaling 组中实例的 CPUUtilization。

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
```

```
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
          }
        ]
      },
      "CustomizedCapacityMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
          }
        ]
      }
    }
  ]
}
```

该示例返回策略的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-
scaling-policy",
  "Alarms": []
}
```

预测性扩展策略中自定义指标的注意事项

如果在使用自定义指标时出现问题，建议您执行以下操作：

- 如果提供了错误消息，请阅读该消息并解决其报告的问题（如果可能）。
- 如果您未事先验证表达式，则该 put-scaling-policy命令会在您创建扩展策略时对其进行验证。但是，此命令有可能无法识别所检测错误的确切原因。要修复这些问题，请对您在get-metric-data命令请求的响应中收到的错误进行故障排除。您也可以从 CloudWatch 控制台对表达式进行故障排除。

- 如果 `MetricDataQueries` 自行指定 SEARCH() 函数，而没有像 SUM() 这样的数学函数，则必须为 `ReturnData` 指定 `false`。原因在于搜索表达式可能返回多个时间序列，而基于表达式的指标规范仅可以返回一个时间序列。
- 搜索表达式中涉及的所有指标均应该具有相同的分辨率。

限制

适用以下限制：

- 您可以在一个指标规范中查询最多 10 个指标的数据点。
- 为满足此限制，一个表达式算作一个指标。

教程：配置自动扩缩以处理繁重的工作负载

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。

内容

前提条件

本教程假定您已执行以下操作：

- 创建了一个 AWS 账户.
- 已安装并配置 AWS CLI。
- 授予了使用 Application Auto Scaling 将资源注册和取消注册为可扩展目标的必要权限。此外，还授予了创建扩展策略和计划操作的必要权限。有关更多信息，请参阅 适用于 Application Auto Scaling 的 Identity and Access Management。
- 创建了在非生产环境中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 AWS 服务和资源的信息，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用 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。为了帮助您解决该问题，输出中将显示一条消息，指示最小容量不能小于指定值，并将提供 AWS 资源可以接受的最小容量值。

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 AWS 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 AWS 服务 可以与 Application Auto Scaling 一起使用 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
  ```

  如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-
id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

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您的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。

   按照指定的计划（世界标准时间每天晚上 8:00），Application Auto Scaling 会按照以下put-scheduled-action命令的指示MaxCapacity将目标的MinCapacity和更新为 0。

   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。

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。换句话说，Application Auto Scaling 在信息不足的情况下会优先考虑可用性。

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。

   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 AWS 服务 可以与 Application Auto Scaling 一起使用。

   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，此命令将返回代表您创建的两个 CloudWatch 警报的 ARNs 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 监控您的扩展活动，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：清理

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。

删除扩展配置不会删除底层 AWS 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

下面的 delete-scheduled-action 命令将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命令删除指定的目标跟踪扩展策略。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

使用如下 deregister-scalable-target 命令来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 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 AWS Command Line Interface 用户指南中的在 AWS CLI中将引号和字符串结合使用。

内容

> **Note**
>
> 有关在 Amazon ECS 部署正在进行时暂停横向扩展过程的说明，请参阅以下文档：《Amazon Elastic Container Service 开发人员指南》中的服务自动扩缩和部署

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展策略关联的缩减活动，而不删除扩展策略或其关联的 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展策略关联的横向扩展活动，而不删除扩展策略或其关联的 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。

暂停和恢复扩展活动

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。有关每项服务的更多信息和示例，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用 中的主题。

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
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

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

将 register-scalable-target 命令与 `--suspended-state` 选项一起使用，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
}
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

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展活动。

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

当您准备好恢复扩展活动时，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
```

```
}
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Application Auto Scaling 的扩展活动

Application Auto Scaling 会监控您的扩展策略的 CloudWatch 指标，并在超过阈值时启动扩展活动。当您手动或按照计划修改可扩展目标的最大大小或最小大小时，它也会启动扩展活动。

进行扩展活动时，Application Auto Scaling 会执行以下操作之一：

- 增加可扩展目标的容量（称为横向扩展）
- 减少可扩展目标的容量（称为横向缩减）

您可以查看过去六周的扩展活动。

按可扩展目标查找扩展活动

要查看特定可扩展目标的扩展活动，请使用以下describe-scaling-activities命令。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-activities --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service
```

Windows

```
aws application-autoscaling describe-scaling-activities --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/my-cluster/my-service
```

在以下示例响应中，StatusCode 包含活动的当前状态，StatusMessage 包含有关扩展活动状态的信息。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/my-cluster/my-service",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
```

```
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
            "EndTime": 1462575872.111,
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered policy
 web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change successfully
 fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

有关响应中字段的描述，请参阅《App lication Auto Scaling API 参考》ScalingActivity中的。

以下状态代码指示引发扩展活动的扩展事件何时达到完成状态：

- `Successful` – 扩展已成功完成
- `Overridden` – 所需容量已由较新的扩展事件进行更新
- `Unfulfilled` – 扩展超时或目标服务无法满足请求
- `Failed` – 扩展失败，出现异常

> **Note**
>
> 扩展活动的状态也可能为 `Pending` 或 `InProgress`。在目标服务响应之前，所有扩展活动都具有 `Pending` 状态。目标响应后，扩展活动的状态更改为 `InProgress`。

包括未扩展的活动

默认情况下，扩展活动不反映 Application Auto Scaling 决定是否不扩展的时间。

例如，假设 Amazon ECS 服务超出给定指标的最大阈值，但任务数已达到允许的最大任务数。在这种情况下，Application Auto Scaling 不会横向扩展所需的任务数。

要在响应中包括未缩放的活动（不是按比例缩放的活动），请在describe-scaling-activities命令中添加`--include-not-scaled-activities`选项。

Linux、macOS 或 Unix

```
aws application-autoscaling describe-scaling-activities --include-not-scaled-activities
 \
  --service-namespace ecs --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/my-cluster/my-service
```

Windows

```
aws application-autoscaling describe-scaling-activities --include-not-scaled-activities
 --service-namespace ecs --scalable-dimension ecs:service:DesiredCount --resource-
id service/my-cluster/my-service
```

> **Note**
>
> 如果此命令引发错误，请确保已将 AWS CLI 本地版本更新到最新版本。

为了确认响应包含未扩展的活动，输出中会显示部分失败的扩展活动的 NotScaledReasons 元素（如果不是全部失败的话）。

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Attempting to scale due to alarm triggered",
            "ResourceId": "service/my-cluster/my-service",
            "ActivityId": "4d759079-a31f-4d0c-8468-504c56e2eecf",
            "StartTime": 1664928867.915,
            "ServiceNamespace": "ecs",
            "Cause": "monitor alarm web-app-cpu-gt-75 in state ALARM triggered policy
 web-app-cpu-gt-75",
            "StatusCode": "Failed",
            "NotScaledReasons": [
                {
                    "Code": "AlreadyAtMaxCapacity",
                    "MaxCapacity": 4
                }
            ]
        }
    ]
}
```

有关响应中字段的描述，请参阅《App lication Auto Scaling API 参考》ScalingActivity中的。

如果返回未扩展的活动，根据 Code 中列出的原因代码，响应中可能会出现 `CurrentCapacity`、`MaxCapacity` 和 `MinCapacity` 等属性。

为防止出现大量重复的条目，只有第一个未扩展的活动才会记录在扩展活动历史记录中。除非不扩展的原因发生变化，否则任何后续的未扩展活动都不会生成新条目。

原因代码

以下是未扩展的活动的原因代码。

原因代码	定义			
AutoScalingAnticipatedFlapping	Auto Scaling 算法决定不采取扩展操作，因为这会导致摆动。摆动是横向缩减和横向扩展的无限循环。也就是说，如果采取扩展操作，则指标值将更改以启动另一个相反方向的扩展操作。			
TargetServicePutResourceAsUnscalable	目标服务暂时将资源置于不可扩展状态。当满足扩展策略中指定的自动扩展条件时，Application Auto Scaling 将尝试再次进行扩展。			

原因代码	定义			
AlreadyAtMaxCapacity	扩展被指定的最大容量阻止。如果您希望 Application Auto Scaling 进行横向扩展，则需要增加最大容量。			
AlreadyAtMinCapacity	扩展被指定的最小容量阻止。如果您希望 Application Auto Scaling 进行横向缩减，则需要减少最小容量。			
AlreadyAtDesiredCapacity	Auto Scaling 算法计算得出，修改后的容量符合当前容量。			

监控 Application Auto Scaling

监控是维护 Application Auto Scaling 和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。您应该从 AWS 解决方案的各个部分收集监控数据，以便在出现多点故障时可以更轻松地进行调试。 AWS 提供监控工具，用于监视 Application Auto Scaling，在出现问题时进行报告，并在适当时自动采取措施。

您可以使用以下功能来帮助您管理 AWS 资源：

AWS CloudTrail

使用 AWS CloudTrail，您可以跟踪您的个人或代表您对 Application Auto Scaling API 进行的调用 AWS 账户。 CloudTrail 将信息存储在您指定的 Amazon S3 存储桶中的日志文件中。您可以标识调用 Application Auto Scaling 的具体用户和账户、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 使用记录应用程序 Auto Scaling API 调用 AWS CloudTrail。

> **Note**
>
> 有关可帮助您记录和收集工作负载数据的其他 AWS 服务的信息，请参阅AWS 规范性指南中的应用程序所有者日志记录和监控指南。

Amazon CloudWatch

Amazon CloudWatch 可帮助您分析日志，并实时监控您的 AWS 资源和托管应用程序的指标。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪资源利用率，并在利用率非常高或指标警报进入`INSUFFICIENT_DATA`状态时通知您。有关更多信息，请参阅 使用监控可扩展资源的使用情况 CloudWatch。

CloudWatch 还会跟踪 Application Auto Scaling 的 AWS API 使用情况指标。您可以使用这些指标来配置警报，以在 API 调用量超过您定义的阈值时提醒您。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的AWS 使用量指标。

Amazon EventBridge

Amazon EventBridge 是一项无服务器事件总线服务，可以轻松地将您的应用程序与来自各种来源的数据连接起来。 EventBridge 提供来自您自己的应用程序、 Software-as-a-Service (SaaS) 应用程序和 AWS 服务的实时数据流，并将这些数据路由到 Lambda 等目标。这让您可以监控服务中

发生的事件，并构建事件驱动型架构。有关更多信息，请参阅 使用 Amazon 监控应用程序 Auto Scaling 事件 EventBridge。

AWS Health Dashboard

Health Dashboard (PHD) 显示信息，还提供 AWS 资源运行状况变化时调用的通知。信息会以两种方式显示：在显示按类别组织的最近和未来事件的控制面板上，以及在显示过去 90 天内所有事件的完整事件日志中。有关更多信息，请参阅 Health Dashboard入门。

使用监控可扩展资源的使用情况 CloudWatch

借助 Amazon CloudWatch，您可以近乎持续地查看可扩展资源中的应用程序。 CloudWatch 是一项 AWS 资源监控服务。您可以使用 CloudWatch 来收集和跟踪指标、设置警报以及自动对 AWS 资源变化做出反应。您还可以创建控制面板来监控所需的特定指标或指标集。

当您与与 Application Auto Scaling 集成的服务进行交互时，它们会将下表所示的指标发送到 CloudWatch。在中 CloudWatch，指标首先按服务命名空间分组，然后按每个命名空间内的各种维度组合进行分组。这些指标可以帮助您监控资源使用量并计划应用程序的容量。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 Auto Scaling。有关这些指标的详细描述，请参阅相关指标的文档。

内容

CloudWatch 用于监控资源使用情况的指标

下表列出了可用于支持监控资源使用情况的 CloudWatch 指标。此列表并不详尽，但能为您提供一个好起点。如果您在 CloudWatch 控制台中看不到这些指标，请确保您已完成资源的设置。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
WorkSpaces 应用程序			

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
车队	AWS/AppStream	姓名：AvailableCapacity 维度：实例集	WorkSpaces 应用程序指标
车队	AWS/AppStream	姓名：CapacityUtilization 维度：实例集	WorkSpaces 应用程序指标
Aurora			
副本	AWS/RDS	姓名：CPUUtilization 尺寸：DBCluster标识符、角色（读者）	Aurora 集群级指标
副本	AWS/RDS	姓名：DatabaseConnections 尺寸：DBCluster标识符、角色（读者）	Aurora 集群级指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
Amazon Comprehend			
文档分类端点	AWS/Comprehend	姓名：InferenceUtilization 维度：EndpointArn	Amazon Comprehend 端点指标
实体识别程序端点	AWS/Comprehend	姓名：InferenceUtilization 维度：EndpointArn	Amazon Comprehend 端点指标
DynamoDB			
表和全局二级索引	AWS/DynamoDB	姓名：ProvisionedReadCapacityUnits 尺寸：TableName，GlobalSecondaryIndexName	DynamoDB 指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
表和全局二级索引	AWS/DynamoDB	姓名：ProvisionedWriteCapacityUnits 尺寸：TableName，GlobalSecondaryIndexName	DynamoDB 指标
表和全局二级索引	AWS/DynamoDB	姓名：ConsumedReadCapacityUnits 尺寸：TableName，GlobalSecondaryIndexName	DynamoDB 指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
表和全局二级索引	AWS/DynamoDB	姓名：ConsumedWriteCapacityUnits 尺寸：TableName，GlobalSecondaryIndexName	DynamoDB 指标
Amazon ECS			
Services	AWS/ECS	姓名：CPUUtilization 尺寸：ClusterName，ServiceName	Amazon ECS 指标
Services	AWS/ECS	姓名：MemoryUtilization 尺寸：ClusterName，ServiceName	Amazon ECS 指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
Services	AWS/ApplicationELB	姓名：RequestCountPerTarget 维度：TargetGroup	应用程序负载均衡器指标
ElastiCache			
集群（复制组）	AWS/ElastiCache	姓名：DatabaseMemoryUsageCountedForEvictPercentage 维度：ReplicationGroupId	ElastiCache Valkey 和 Redis OSS 指标
集群（复制组）	AWS/ElastiCache	姓名：DatabaseCapacityUsageCountedForEvictPercentage 维度：ReplicationGroupId	ElastiCache Valkey 和 Redis OSS 指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
集群（复制组）	AWS/ElastiCache	名称：发动机 CPUUtilization 维度：ReplicationGroupId，角色（主要）	ElastiCache Valkey 和 Redis OSS 指标
集群（复制组）	AWS/ElastiCache	名称：发动机 CPUUtilization 尺寸：ReplicationGroupId，角色（复制品）	ElastiCache Valkey 和 Redis OSS 指标
集群（缓存）	AWS/ElastiCache	名称：发动机 CPUUtilization 尺寸：CacheClusterId，节点	ElastiCache 内存缓存指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
集群（缓存）	AWS/ElastiCache	姓名：DatabaseCapacityMemoryUsagePercentage 尺寸：CacheClusterId	ElastiCache 内存缓存指标
Amazon EMR			
集群	AWS/ElasticMapReduce	姓名：YARNMemoryAvailablePercentage 维度：ClusterId	Amazon EMR 指标
Amazon Keyspaces			
表	AWS/Cassandra	姓名：ProvisionedReadCapacityUnits 尺寸：Keyspace，TableName	Amazon Keyspaces 指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
表	AWS/Cassandra	姓名：ProvisionedWriteCapacityUnits 尺寸：Keyspace，TableName	Amazon Keyspaces 指标
表	AWS/Cassandra	姓名：ConsumedReadCapacityUnits 尺寸：Keyspace，TableName	Amazon Keyspaces 指标
表	AWS/Cassandra	姓名：ConsumedWriteCapacityUnits 尺寸：Keyspace，TableName	Amazon Keyspaces 指标
Lambda			

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
预配置并发	AWS/Lambda	姓名：ProvisionedConcurrencyUtilization 尺寸：FunctionName，资源	Lambda 函数指标
Amazon MSK			
代理存储	AWS/Kafka	姓名：KafkaDataLogsDiskUsed 维度：集群名称	Amazon MSK 指标
代理存储	AWS/Kafka	姓名：KafkaDataLogsDiskUsed 维度：集群名称、代理 ID	Amazon MSK 指标
Neptune			

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
集群	AWS/Neptune	姓名：CPUUtilization 尺寸：DBCluster 标识符、角色（读者）	Neptune 指标
SageMaker AI			
端点变体	AWS/SageMaker	姓名：InvocationsPerInstance 尺寸：EndpointName，VariantName	调用指标
推理组件	AWS/SageMaker	姓名：InvocationsPerCopy 尺寸：InferenceComponentName	调用指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
无服务器端点的预置并发	AWS/SageMaker	姓名：ServerlessProvisionedConcurrencyUtilization 尺寸：EndpointName，VariantName	无服务器端点指标
Spot 实例集（Amazon EC2）			
Spot Fleets	AWS/EC2Spot	姓名：CPUUtilization 维度：FleetRequestId	竞价型实例集指标
Spot Fleets	AWS/EC2Spot	姓名：NetworkIn 维度：FleetRequestId	竞价型实例集指标

可扩展资源	命名空间	CloudWatch 指标	指向文档的链接
Spot Fleets	AWS/EC2Spot	姓名：NetworkOut 维度：FleetRequestId	竞价型实例集指标
Spot Fleets	AWS/ApplicationELB	姓名：RequestCountPerTarget 维度：TargetGroup	应用程序负载均衡器指标

目标跟踪扩展策略的预定义目标

下表列出了《App lication Auto Scaling API 参考》中的预定义指标类型及其相应的 CloudWatch 指标名称。每个预定义的指标都代表基础 CloudWatch 指标值的聚合。除非另有说明，否则结果是一分钟内基于百分比的平均资源使用量。预定义指标仅在设置目标跟踪扩展策略的情况下使用。

有关这些指标的更多信息，请参阅 CloudWatch 用于监控资源使用情况的指标 中的表格内可用的服务文档。

预定义指标类型	CloudWatch 指标名称
WorkSpaces 应用程序	
`AppStreamAverageCapacityUtilization`	CapacityUtilization
Aurora	

预定义指标类型	CloudWatch 指标名称
RDSReaderAverageCPUUtilization	CPUUtilization
RDSReaderAverageDatabaseConnections	DatabaseConnections[1]
Amazon Comprehend	
ComprehendInferenceUtilization	InferenceUtilization
DynamoDB	
DynamoDBReadCapacityUtilization	ProvisionedReadCapacityUnits，ConsumedReadCapacityUnits[2]
DynamoDBWriteCapacityUtilization	ProvisionedWriteCapacityUnits，ConsumedWriteCapacityUnits[2]
Amazon ECS	
ECSServiceAverageCPUUtilization	CPUUtilization
ECSServiceAverageMemoryUtilization	MemoryUtilization
ALBRequestCountPerTarget	RequestCountPerTarget[1]
ElastiCache	
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage
ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage	DatabaseCapacityUsageCountedForEvictPercentage
ElastiCachePrimaryEngineCPUUtilization	发动机 CPUUtilization
ElastiCacheReplicaEngineCPUUtilization	发动机 CPUUtilization

预定义指标类型	CloudWatch 指标名称
ElastiCacheEngineCPUUtilization	发动机 CPUUtilization
ElastiCacheDatabaseMemoryUsagePercentage	DatabaseMemoryUsagePercentage
Amazon Keyspaces	
CassandraReadCapacityUtilization	ProvisionedReadCapacityUnits， ConsumedReadCapacityUnits[2]
CassandraWriteCapacityUtilization	ProvisionedWriteCapacityUnits， ConsumedWriteCapacityUnits[2]
Lambda	
LambdaProvisionedConcurrencyUtilization	ProvisionedConcurrencyUtilization
Amazon MSK	
KafkaBrokerStorageUtilization	KafkaDataLogsDiskUsed
Neptune	
NeptuneReaderAverageCPUUtilization	CPUUtilization
SageMaker AI	
SageMakerVariantInvocationsPerInstance	InvocationsPerInstance[1]
SageMakerInferenceComponentInvocationsPerCopy	InvocationsPerCopy[1]
SageMakerVariantProvisionedConcurrencyUtilization	ServerlessProvisionedConcurrencyUtilization

预定义指标类型	CloudWatch 指标名称
SageMakerInferenceComponentConcurrentRequestsPerCopyHighResolution	ConcurrentRequestsPerCopy
SageMakerVariantConcurrentRequestsPerModelHighResolution	ConcurrentRequestsPerModel
竞价型实例集	
EC2SpotFleetRequestAverageCPUUtilization	CPUUtilization[3]
EC2SpotFleetRequestAverageNetworkIn[3]	NetworkIn[1]
EC2SpotFleetRequestAverageNetworkOut[3]	NetworkOut[1]
ALBRequestCountPerTarget	RequestCountPerTarget[1]

[1] 指标基于计数，而不是百分比。

[2] 对于 DynamoDB 和 Amazon Keyspaces，预定义指标是两个指标 CloudWatch的汇总，以支持根据预配置的吞吐量消耗进行扩展。

[3] 为了获得最佳扩展性能，应使用 Amazon EC2 详细监控。

预测性扩缩指标和维度

AWS/ApplicationAutoScaling命名空间包括以下预测性扩展策略指标。这些指标的分辨率为一小时，通过将预测值与实际值进行比较，可以帮助您评估预测的准确性。

指标	说明	Dimensions
PredictiveScalingLoadForecast	应用程序预计将生成的负载量。	ResourceId，ServiceNamespace，

指标	说明	Dimensions
	Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	PolicyName，ScalableDimension，PairIndex
PredictiveScalingCapacityForecast	满足应用程序需求所需的预期容量。这是基于负载预测和目标利用率水平，您要在该水平上维护 Application Auto Scaling 资源。 Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	ResourceId，ServiceNamespace，PolicyName，ScalableDimension
PredictiveScalingMetricPairCorrelation	扩展指标与负载指标的每个实例平均值之间的相关性。预测性扩展假设相关性很高。因此，如果您观察到该指标的值很低，最好不要使用指标对。 Average、Minimum 和 Maximum 统计数据非常有用，而 Sum 统计数据用处不大。 报告标准：在创建初始预测后报告。	ResourceId，ServiceNamespace，PolicyName，ScalableDimension，PairIndex

使用记录应用程序 Auto Scaling API 调用 AWS CloudTrail

Application Auto Scaling 与AWS CloudTrail一项服务集成，该服务提供用户、角色或角色所执行操作的记录 AWS 服务。 CloudTrail 将 Application Auto Scaling 的 API 调用捕获为事件。所捕获的调用包含来自 AWS 管理控制台 的调用，以及对 Application Auto Scaling API 操作的代码调用。使用收集的信息 CloudTrail，您可以确定向 Application Auto Scaling 发出的请求、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。

- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

CloudTrail 在您创建账户 AWS 账户 时在您的账户中处于活动状态，并且您可以自动访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 AWS 区域有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 AWS 账户 过去 90 天内的事件，请创建跟踪。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 AWS 管理控制台 都是多区域的。您可以通过使用 AWS CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可以捕获账户 AWS 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

中的应用程序 Auto Scaling 管理事件 CloudTrail

管理事件提供有关对中的资源执行的管理操作的信息 AWS 账户。这些也称为控制面板操作。默认情况下， CloudTrail 记录管理事件。

Application Auto Scaling 将所有 Application Auto Scaling 控制面板操作记录为管理事件。有关 Application Auto Scaling 记录到的应用程序 Auto Scaling 控制平面操作的列表 CloudTrail，请参阅《应用程序自动缩放 API 参考》。

Application Auto Scaling 事件示例

事件代表来自任何来源的单个请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序出现。

以下示例显示了一个演示该`DescribeScalableTargets`操作 CloudTrail 的事件。

```
{
```

```
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
    },
    "eventTime": "2018-08-16T23:20:32Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "DescribeScalableTargets",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "72.21.196.68",
    "userAgent": "EC2 Spot Console",
    "requestParameters": {
        "serviceNamespace": "ec2",
        "scalableDimension": "ec2:spot-fleet-request:TargetCapacity",
        "resourceIds": [
            "spot-fleet-request/sfr-05ceaf79-3ba2-405d-e87b-612857f1357a"
        ]
    },
    "responseElements": null,
    "additionalEventData": {
        "service": "application-autoscaling"
    },
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


有关 CloudTrail 录音内容的信息，请参阅《AWS CloudTrail 用户指南》中的CloudTrail录制内容。

应用程序 Auto Scaling RemoveAction 正在调用 CloudWatch

您的 AWS CloudTrail 日志可能会显示，当 Application Auto Scaling 指示从警报中删除自动扩展操作时，Appl CloudWatch ication Auto Scaling 会调用 CloudWatch `RemoveAction` API。如果取消注册了可扩缩目标、删除了扩缩策略或警报调用了不存在的扩缩策略，都有可能发生这种情况。

使用 Amazon 监控应用程序 Auto Scaling 事件 EventBridge

Amazon EventBridge（以前称为 CloudWatch 事件）可帮助您监控特定于 Application Auto Scaling 的事件，并启动使用其他事件的目标操作 AWS 服务。来自 AWS 服务 的事件以近乎实时 EventBridge 的方式传送到。

使用 EventBridge，您可以创建匹配传入事件的规则，并将它们路由到目标进行处理。

有关更多信息，请参阅《亚马逊 EventBridge 用户指南》 EventBridge中的 “亚马逊入门”。

Application Auto Scaling 事件

以下是 Application Auto Scaling 的示例事件。事件会尽可能生成。

Application Auto Scaling 目前只有特定于缩放到最大值的事件和 API 调用通过 CloudTrail 的事件才可用。

事件类型

- 状态更改的事件：扩展到最大容量
- 通过以下方式进行的 API 调用事件 CloudTrail

状态更改的事件：扩展到最大容量

以下示例事件显示 Application Auto Scaling 将可扩展目标的容量增加（横向扩展）至其最大容量限制。如果需求再次增加，Application Auto Scaling 将无法进一步扩展目标，因为它已经扩展到其最大容量。

在 `detail` 对象中，`resourceId`、`serviceNamespace` 和 `scalableDimension` 属性的值用来标识可扩展目标。`newDesiredCapacity` 和 `oldDesiredCapacity` 属性的值用于指横向扩展事件完成后的新容量和横向扩展事件开始前的原始容量。`maxCapacity` 是可伸缩目标的最大容量限制。

```
{
  "version": "0",
  "id": "11112222-3333-4444-5555-666677778888",
```

```
  "detail-type": "Application Auto Scaling Scaling Activity State Change",
  "source": "aws.application-autoscaling",
  "account": "123456789012",
  "time": "2019-06-12T10:23:40Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "startTime": "2022-06-12T10:20:43Z",
    "endTime": "2022-06-12T10:23:40Z",
    "newDesiredCapacity": 8,
    "oldDesiredCapacity": 5,
    "minCapacity": 2,
    "maxCapacity": 8,
    "resourceId": "table/my-table",
    "scalableDimension": "dynamodb:table:WriteCapacityUnits",
    "serviceNamespace": "dynamodb",
    "statusCode": "Successful",
    "scaledToMax": true,
    "direction": "scale-out"
}
```

要创建捕获所有可扩展目标的所有 `scaledToMax` 状态更改事件的规则，请使用以下示例事件模式。

```
{
  "source": [
    "aws.application-autoscaling"
  ],
  "detail-type": [
    "Application Auto Scaling Scaling Activity State Change"
  ],
  "detail": {
    "scaledToMax": [
      true
    ]
  }
}
```

通过以下方式进行的 API 调用事件 CloudTrail

跟踪是一种 AWS CloudTrail 用于将事件作为日志文件传输到 Amazon S3 存储桶的配置。 CloudTrail 日志文件包含日志条目。一个事件代表一个日志条目，包括有关所请求操作的信息、操作的日期和时间以及请求参数。要了解如何开始使用 CloudTrail，请参阅AWS CloudTrail 用户指南中的创建跟踪。

通过传递的事件 CloudTrail `AWS API Call via CloudTrail`具有的价值`detail-type`。

以下示例事件表示一个 CloudTrail 日志文件条目，该条目显示控制台用户调用了 Application Auto Scaling RegisterScalableTarget操作。

```
{
  "version": "0",
  "id": "99998888-7777-6666-5555-444433332222",
  "detail-type": "AWS API Call via CloudTrail",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-07-13T16:50:15Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "eventVersion": "1.08",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "123456789012",
      "arn": "arn:aws:iam::123456789012:user/Bob",
      "accountId": "123456789012",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "123456789012",
          "arn": "arn:aws:iam::123456789012:role/Admin",
          "accountId": "123456789012",
          "userName": "Admin"
        },
        "webIdFederationData": {},
        "attributes": {
          "creationDate": "2022-07-13T15:17:08Z",
          "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2022-07-13T16:50:15Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "RegisterScalableTarget",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "EC2 Spot Console",
    "requestParameters": {
```

```
      "resourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "serviceNamespace": "ec2",
      "scalableDimension": "ec2:spot-fleet-request:TargetCapacity",
      "minCapacity": 2,
      "maxCapacity": 10
    },
    "responseElements": null,
    "additionalEventData": {
      "service": "application-autoscaling"
    },
    "requestID": "e9caf887-8d88-11e5-a331-3332aa445952",
    "eventID": "49d14f36-6450-44a5-a501-b0fdcdfaeb98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
  }
}
```

要基于所有可扩展目标的全部DeleteScalingPolicy和 DeregisterScalableTargetAPI 调用创建规则，请使用以下示例事件模式：

```
{
  "source": [
    "aws.autoscaling"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "autoscaling.amazonaws.com"
    ],
    "eventName": [
       "DeleteScalingPolicy",
       "DeregisterScalableTarget"
    ],
    "additionalEventData": {
      "service": [
        "application-autoscaling"
      ]
```

```
    }
  }
}
```

有关使用的更多信息 CloudTrail，请参阅使用记录应用程序 Auto Scaling API 调用 AWS CloudTrail。

将此服务与 AWS SDK 配合使用

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK 代码示例
适用于 Kotlin 的 AWS SDK	适用于 Kotlin 的 AWS SDK 代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK 代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK 代码示例
AWS Tools for PowerShell	AWS Tools for PowerShell 代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK 代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK 代码示例
适用于 Rust 的 AWS SDK	适用于 Rust 的 AWS SDK 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
适用于 Swift 的 AWS SDK	适用于 Swift 的 AWS SDK 代码示例

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

Application Auto Scaling 的代码示例 AWS SDKs

以下代码示例展示了如何将 Application Auto Scaling 与 AWS 软件开发套件 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- Application Auto Scaling 的基本示例 AWS SDKs
  - Application Auto Scaling 的操作使用于 AWS SDKs
    - DeleteScalingPolicy与 AWS SDK 或 CLI 配合使用
    - 将 DeleteScheduledAction 与 CLI 配合使用
    - 将 DeregisterScalableTarget 与 CLI 配合使用
    - 将 DescribeScalableTargets 与 CLI 配合使用
    - 将 DescribeScalingActivities 与 CLI 配合使用
    - DescribeScalingPolicies与 AWS SDK 或 CLI 配合使用
    - 将 DescribeScheduledActions 与 CLI 配合使用
    - 将 PutScalingPolicy 与 CLI 配合使用
    - 将 PutScheduledAction 与 CLI 配合使用
    - RegisterScalableTarget与 AWS SDK 或 CLI 配合使用

Application Auto Scaling 的基本示例 AWS SDKs

以下代码示例展示了如何使用 Application Auto Scaling 的基础知识 AWS SDKs。

示例

- Application Auto Scaling 的操作使用于 AWS SDKs
  - DeleteScalingPolicy与 AWS SDK 或 CLI 配合使用
  - 将 DeleteScheduledAction 与 CLI 配合使用
  - 将 DeregisterScalableTarget 与 CLI 配合使用
  - 将 DescribeScalableTargets 与 CLI 配合使用

- 将 DescribeScalingActivities 与 CLI 配合使用
- DescribeScalingPolicies与 AWS SDK 或 CLI 配合使用
- 将 DescribeScheduledActions 与 CLI 配合使用
- 将 PutScalingPolicy 与 CLI 配合使用
- 将 PutScheduledAction 与 CLI 配合使用
- RegisterScalableTarget与 AWS SDK 或 CLI 配合使用

Application Auto Scaling 的操作使用于 AWS SDKs

以下代码示例演示了如何使用执行各个 Application Auto Scaling 操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Application Auto Scaling API 参考。

示例

- DeleteScalingPolicy与 AWS SDK 或 CLI 配合使用
- 将 DeleteScheduledAction 与 CLI 配合使用
- 将 DeregisterScalableTarget 与 CLI 配合使用
- 将 DescribeScalableTargets 与 CLI 配合使用
- 将 DescribeScalingActivities 与 CLI 配合使用
- DescribeScalingPolicies与 AWS SDK 或 CLI 配合使用
- 将 DescribeScheduledActions 与 CLI 配合使用
- 将 PutScalingPolicy 与 CLI 配合使用
- 将 PutScheduledAction 与 CLI 配合使用
- RegisterScalableTarget与 AWS SDK 或 CLI 配合使用

`DeleteScalingPolicy`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteScalingPolicy`。

CLI

AWS CLI

删除扩展策略

此示例删除在默认集群中运行的 Amazon ECS 服务 web-app 的扩展策略。

命令:

```
aws application-autoscaling delete-scaling-policy --policy-name web-app-cpu-lt-25
 --scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-
app --service-namespace ecs
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteScalingPolicy中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReq
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
```

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
```

```
   }

   public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
       try {
           DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
               .policyName(policyName)
               .scalableDimension(tableWCUs)
               .serviceNamespace(ns)
               .resourceId(tableId)
               .build();

           appAutoScalingClient.deleteScalingPolicy(delSPRequest);
           System.out.println(policyName +" was deleted successfully.");

       } catch (ApplicationAutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
   }

   // Verify that the scaling policy was deleted
   public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
           .scalableDimension(tableWCUs)
           .serviceNamespace(ns)
           .resourceId(tableId)
           .build();

       DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
       System.out.println("DescribeScalableTargets result: ");
       System.out.println(response);
   }

   public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
       try {
```

```
            DeregisterScalableTargetRequest targetRequest =
 DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteScalingPolicy中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定扩展策略。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DeleteScalingPolicy中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定扩展策略。

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DeleteScalingPolicy中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteScheduledAction** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteScheduledAction。

CLI

AWS CLI

删除计划操作

以下delete-scheduled-action示例从指定的 Amazon AppStream 2.0 队列中删除指定的计划操作：

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace appstream \
    --scalable-dimension appstream:fleet:DesiredCapacity \
    --resource-id fleet/sample-fleet \
    --scheduled-action-name my-recurring-action
```

此命令不生成任何输出。

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的计划扩展。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteScheduledAction中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定计划操作。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DeleteScheduledAction中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 删除 Application Auto Scaling 可扩展目标的指定计划操作。

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DeleteScheduledAction中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeregisterScalableTarget** 与 CLI 配合使用

以下代码示例演示如何使用 `DeregisterScalableTarget`。

CLI

AWS CLI

取消注册可扩展目标

此示例取消注册在默认集群中运行的 Amazon ECS 服务（名为 web-app）的可扩展目标。

命令:

```
aws application-autoscaling deregister-scalable-target --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app
```

此示例取消注册自定义资源的可扩展目标。 custom-resource-id.txt 文件包含一个用于标识资源 ID 的字符串，对于自定义资源，该字符串是通过您的 Amazon API Gateway 终端节点指向自定义资源的路径。

命令:

```
aws application-autoscaling deregister-scalable-target --
service-namespace custom-resource --scalable-dimension custom-
resource:ResourceType:Property --resource-id file://~/custom-resource-id.txt
```

custom-resource-id.txt 文件的内容：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeregisterScalableTarget中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 取消注册 Application Auto Scaling 可扩展目标。取消注册可扩展目标会删除与其关联的扩展策略。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DeregisterScalableTarget中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 取消注册 Application Auto Scaling 可扩展目标。取消注册可扩展目标会删除与其关联的扩展策略。

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DeregisterScalableTarget中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeScalableTargets** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeScalableTargets。

CLI

AWS CLI

描述可扩展目标

以下 describe-scalable-targets 示例描述 ecs 服务命名空间的可扩展目标。

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

输出：

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "ecs",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "MinCapacity": 1,
            "MaxCapacity": 10,
            "RoleARN": "arn:aws:iam::123456789012:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService",
            "CreationTime": 1462558906.199,
            "SuspendedState": {
                "DynamicScalingOutSuspended": false,
                "ScheduledScalingSuspended": false,
                "DynamicScalingInSuspended": false
            },
            "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
        }
    ]
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的可与 Application Auto Scaling 一起使用的AWS 服务。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeScalableTargets中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将提供有关指定命名空间中 Application Auto Scaling 可扩展目标的信息。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

输出：

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DescribeScalableTargets中的。

适用于 PowerShell V5 的工具

示例 1：此示例将提供有关指定命名空间中 Application Auto Scaling 可扩展目标的信息。

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

输出：

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
```

```
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DescribeScalableTargets中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeScalingActivities** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeScalingActivities。

CLI

AWS CLI

示例 1：描述指定 Amazon ECS 服务的扩展活动

以下 describe-scaling-activities 示例描述在 default 集群中运行的 Amazon ECS 服务（名为 web-app）的扩展活动。输出显示了由扩展策略启动的扩展活动。

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace ecs \
    --resource-id service/default/web-app
```

输出：

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/default/web-app",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
```

```
            "EndTime": 1462575872.111,
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered
 policy web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change
 successfully fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的 Application Auto Scaling 的扩展活动。

示例 2：描述指定 DynamoDB 表的扩展活动

以下 `describe-scaling-activities` 示例描述名为 `TestTable` 的 DynamoDB 表的扩展活动。输出显示了由两个不同的计划操作启动的扩展活动。

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace dynamodb \
    --resource-id table/TestTable
```

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"
        }
    ]
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的 Application Auto Scaling 的扩展活动。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeScalingActivities中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：提供有关前六周指定命名空间中扩展活动的描述性信息。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

输出：

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DescribeScalingActivities中的。

适用于 PowerShell V5 的工具

示例 1：提供有关前六周指定命名空间中扩展活动的描述性信息。

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

输出：

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
```

```
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DescribeScalingActivities中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeScalingPolicies`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeScalingPolicies`。

CLI

AWS CLI

描述扩展策略

此示例命令描述 ecs 服务命名空间的扩展策略。

命令:

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

输出：

```
{
    "ScalingPolicies": [
        {
            "PolicyName": "web-app-cpu-gt-75",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462561899.23,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 60,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": 200,
```

```
                    "MetricIntervalLowerBound": 0.0
                }
            ],
            "AdjustmentType": "PercentChangeInCapacity"
        },
        "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-gt-75",
        "PolicyType": "StepScaling",
        "Alarms": [
            {
                "AlarmName": "web-app-cpu-gt-75",
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-gt-75"
            }
        ],
        "ServiceNamespace": "ecs"
    },
    {
        "PolicyName": "web-app-cpu-lt-25",
        "ScalableDimension": "ecs:service:DesiredCount",
        "ResourceId": "service/default/web-app",
        "CreationTime": 1462562575.099,
        "StepScalingPolicyConfiguration": {
            "Cooldown": 1,
            "StepAdjustments": [
                {
                    "ScalingAdjustment": -50,
                    "MetricIntervalUpperBound": 0.0
                }
            ],
            "AdjustmentType": "PercentChangeInCapacity"
        },
        "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-lt-25",
        "PolicyType": "StepScaling",
        "Alarms": [
            {
                "AlarmName": "web-app-cpu-lt-25",
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-lt-25"
            }
        ],
```

```
            "ServiceNamespace": "ecs"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeScalingPolicies](#)中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 描述了指定服务命名空间的 Application Auto Scaling 扩展策略。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

输出：

```
Alarms                                    : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                                : default-scale-out
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                                : default-scale-in
PolicyType                                : StepScaling
```

```
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DescribeScalingPolicies中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 描述了指定服务命名空间的 Application Auto Scaling 扩展策略。

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

输出：

```
Alarms                                    : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                                : default-scale-out
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                                : default-scale-in
PolicyType                                : StepScaling
```

```
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DescribeScalingPolicies中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeScalingPolicies于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DescribeScheduledActions** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeScheduledActions。

CLI

AWS CLI

描述计划的操作

以下 describe-scheduled-actions 示例显示指定服务命名空间的计划操作的详细信息：

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

输出：

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table: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/my-table",
            "CreationTime": 1561571946.021,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
```

```
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-second-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 5,
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-second-scheduled-action",
            "ServiceNamespace": "dynamodb"
        }
    ]
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的计划扩展。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeScheduledActions中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 列出了为自动扩缩组计划但尚未运行的操作或尚未达到其结束时间的操作。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

输出：

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DescribeScheduledActions中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 列出了为自动扩缩组计划但尚未运行的操作或尚未达到其结束时间的操作。

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

输出：

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DescribeScheduledActions中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutScalingPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 PutScalingPolicy。

CLI

AWS CLI

示例 1：应用具有预定义指标规范的目标跟踪扩展策略

以下 put-scaling-policy 示例将具有预定义指标规范的目标跟踪扩展策略应用于默认集群中名为 web-app 的 Amazon ECS 服务。该策略将服务的平均 CPU 利用率保持在 75%，横向

扩展和横向缩减冷却时间为 60 秒。输出包含代表您创建的两个 CloudWatch 警报的 ARNs 和名称。

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cpu75-target-tracking-scaling-policy --policy-
type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

此示例假设您在当前目录中有一个 config.json 文件，其中包含以下内容：

```
{
     "TargetValue": 75.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

输出：

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/cpu75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
        }
```

```
    ]
}
```

示例 2：应用具有自定义指标规范的目标跟踪扩展策略

以下 put-scaling-policy 示例将具有自定义指标规范的目标跟踪扩展策略应用于默认集群中名为 web-app 的 Amazon ECS 服务。该策略将服务的平均利用率保持在 75%，横向扩展和横向缩减冷却时间为 60 秒。输出包含代表您创建的两个 CloudWatch 警报的 ARNs 和名称。

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cms75-target-tracking-scaling-policy
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

此示例假设您在当前目录中有一个 config.json 文件，其中包含以下内容：

```
{
    "TargetValue":75.0,
    "CustomizedMetricSpecification":{
        "MetricName":"MyUtilizationMetric",
        "Namespace":"MyNamespace",
        "Dimensions": [
            {
                "Name":"MyOptionalMetricDimensionName",
                "Value":"MyOptionalMetricDimensionValue"
            }
        ],
        "Statistic":"Average",
        "Unit":"Percent"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

输出：

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/default/web-
app:policyName/cms75-target-tracking-scaling-policy",
```

```
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

示例 3：为仅向外扩展应用目标跟踪扩展策略

以下 put-scaling-policy 示例将目标跟踪扩展策略应用于默认集群中名为 web-app 的 Amazon ECS 服务。当来自应用程序负载均衡器的 RequestCountPerTarget 指标超过阈值时，该策略用于横向扩展 ECS 服务。输出包含代表您创建的 CloudWatch 警报的 ARN 和名称。

```
aws application-autoscaling put-scaling-policy \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --policy-name alb-scale-out-target-tracking-scaling-policy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

config.json 的内容：

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ALBRequestCountPerTarget",
         "ResourceLabel": "app/EC2Co-EcsEl-1TKLTMITMM0EO/f37c06a68c1748aa/
targetgroup/EC2Co-Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
     },
```

```
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60,
    "DisableScaleIn": true
}
```

输出：

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/alb-scale-out-target-tracking-scaling-
policy",
    "Alarms": [
        {
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        }
    ]
}
```

有关更多信息，请参阅《AWS Application Auto Scaling 用户指南》中的 Application Auto Scaling 的目标跟踪扩展策略。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutScalingPolicy中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新策略。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

输出：

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) PutScalingPolicy中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新策略。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

输出：

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) PutScalingPolicy中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutScheduledAction** 与 CLI 配合使用

以下代码示例演示如何使用 PutScheduledAction。

CLI

AWS CLI

向 DynamoDB 表添加计划操作

此示例将计划操作添加到 DynamoDB 表中，该表 TestTable 名为按周期性计划进行横向扩展。按照指定的计划（世界标准时间每天下午 12:15），如果当前容量低于为 MinCapacity的指定值，Application Auto Scaling 会扩展到指定的值。 MinCapacity

命令:

```
aws application-autoscaling put-scheduled-action --service-
namespace dynamodb --scheduled-action-name my-recurring-action --
schedule "cron(15 12 * * ? *)" --resource-id table/TestTable --
scalable-dimension dynamodb:table:WriteCapacityUnits --scalable-target-
action MinCapacity=6
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的“计划扩展”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutScheduledAction中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新计划操作。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) PutScheduledAction中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 为 Application Auto Scaling 可扩展目标创建或更新计划操作。每个可扩展目标均由服务命名空间、资源 ID 和可扩展维度标识。

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) PutScheduledAction中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`RegisterScalableTarget`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RegisterScalableTarget`。

CLI

AWS CLI

示例 1：将 ECS 服务注册为可扩展目标

以下 `register-scalable-target` 示例向 Application Auto Scaling 注册 Amazon ECS 服务。它还向可扩展目标添加一个带有键名称 `environment` 和值 `production` 的标签。

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

输出：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关其他 AWS 服务和自定义资源的示例，请参阅《Application Auto Scaling 用户指南》中可与 Appl AWS ication Auto Scaling 配合使用的服务中的主题。

示例 2：暂停可扩展目标的扩展活动

以下 register-scalable-target 示例暂停现有可扩展目标的扩展活动。

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=true,DynamicScalingOutSuspended=true,ScheduledScalingSusp
```

输出：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的暂停和恢复 Application Auto Scaling 扩展。

示例 3：恢复可扩展目标的扩展活动

以下 register-scalable-target 示例恢复现有可扩展目标的扩展活动。

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=false,DynamicScalingOutSuspended=false,ScheduledScalingSu
```

输出：

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

有关更多信息，请参阅《Application Auto Scaling 用户指南》中的暂停和恢复 Application Auto Scaling 扩展。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RegisterScalableTarget中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecificati
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetReque
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicy
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
               policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table,
 which is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
```

```
   }

   // Configure a scaling policy.
   public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
       // Check if the policy exists before creating a new one.
       DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
           .serviceNamespace(ns)
           .resourceId(tableId)
           .scalableDimension(tableWCUs)
           .build());

       if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
           // If policies exist, consider updating an existing policy instead of
creating a new one.
           System.out.println("Policy already exists. Consider updating it
instead.");
           List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
           for (ScalingPolicy pol : polList) {
               System.out.println("Policy name:" +pol.policyName());
           }
       } else {
           // If no policies exist, proceed with creating a new policy.
           PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
               .build();

           TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
               .predefinedMetricSpecification(specification)
               .targetValue(50.0)
               .scaleInCooldown(60)
               .scaleOutCooldown(60)
               .build();

           PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
               .targetTrackingScalingPolicyConfiguration(policyConfiguration)
               .serviceNamespace(ns)
```

```
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling
 policy for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " +
 e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RegisterScalableTarget]中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此 cmdlet 注册或更新可扩展目标。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RegisterScalableTarget]中的。

适用于 PowerShell V5 的工具

示例 1：此 cmdlet 注册或更新可扩展目标。可扩展目标是 Application Auto Scaling 可以横向扩展或横向缩减的资源。

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) RegisterScalableTarget中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Application Auto Scaling 的标签支持

您可以使用 AWS CLI 或 SDK 来标记 Application Auto Scaling 可扩展目标。可扩展目标是代表 Application Auto Scaling 可以扩展的 AWS 或自定义资源的实体。

每个标签都包含游湖使用 Application Auto Scaling API 定义的键和值。标签可以帮助您根据组织的需求配置对特定可扩展目标的精细访问权限。有关更多信息，请参阅 结合使用 ABAC 与 Application Auto Scaling。

您可以在注册新的可扩展目标时向目标添加标签，也可以将向现有的可扩展目标添加标签。

用于管理标签的常用命令包括：

- register-scalable-target在注册新的可扩展目标时对其进行标记。
- tag-resource，用于向现有的可扩展目标添加标签。
- list-tags-for-resource返回可扩展目标上的标签。
- untag-resource，用于删除标签。

标签示例

使用以下带`--tags`选项的register-scalable-target命令。该示例可用两个标签来标记可扩展目标：一个名称为 **`environment`**、标签值为 **`production`** 的标签键；一个名称为 **`iscontainerbased`**、标签值为 **`true`** 的标签键。

将和的示例值`--min-capacity``--max-capacity`和的示例文本替换为您在 Application Auto Scaling 中使用的 AWS 服务的命名空间、`--scalable-dimension`与您正在注册的资源关联的可扩展维度以及`--resource-id`资源的标识符。`--service-namespace`有关每项服务的更多信息和示例，请参阅 AWS 服务 可以与 Application Auto Scaling 一起使用 中的主题。

```
aws application-autoscaling register-scalable-target \
  --service-namespace namespace \
  --scalable-dimension dimension \
  --resource-id identifier \
  --min-capacity 1 --max-capacity 10 \
  --tags environment=production,iscontainerbased=true
```

如果成功，该命令会返回可扩展目标的 ARN。

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:region:account-id:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

> **Note**
>
> 如果此命令引发错误，请确保已将 AWS CLI 本地版本更新到最新版本。

安全性标签

使用标签验证请求者（例如 IAM 用户或角色）是否有权限执行某些操作。使用下面的一个或多个条件键，在 IAM policy 的条件元素中提供标签信息：

- 使用 `aws:ResourceTag/tag-key: tag-value` 可允许（或拒绝）带特定标签的可扩展目标上的用户操作。
- 使用 `aws:RequestTag/tag-key: tag-value` 要求在请求中存在（或不存在）特定标签。
- 使用 `aws:TagKeys [tag-key, ...]` 要求在请求中存在（或不存在）特定标签键。

例如，以下 IAM policy 授予执行 `DeregisterScalableTarget`、`DeleteScalingPolicy` 和 `DeleteScheduledAction` 操作的权限。但如果对其执行操作的可扩展目标组具有标签 **`environment=production`**，在此策略也会拒绝操作。

JSON

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

```
        {
            "Effect": "Deny",
            "Action": [
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:DeleteScheduledAction"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "production"
                }
            }
        }
    ]
}
```

控制对标签的访问

使用标签来验证请求者（例如 IAM 用户或角色）是否有权限添加、修改或删除可扩展目标的标签。

例如，您可以创建一个 IAM policy，以仅允许从可扩展目标中删除具有 **temporary** 键的标签。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "application-autoscaling:UntagResource",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
            }
        }
    ]
}
```

Application Auto Scaling 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Appl AWS ication Auto Scaling 的合规计划，请按合规计划查看。
- 云端安全-您的责任由您使用的 AWS 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。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Application Auto Scaling 资源。

内容

Application Auto Scaling 中的数据保护

分担责任模型 AWS 分担责任模型适用于 Application Auto Scaling 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 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。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 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 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 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 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括当你使用控制台、API 或 AWS 服务 来使用 Application Auto Scaling 或其他方式时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本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，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

适用于 Application Auto Scaling 的 Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Application Auto Scaling 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

有关完整的 IAM 文档，请参阅 IAM 用户指南。

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。

以下各节详细介绍 IAM 管理员如何使用 IAM 通过控制谁可以执行 Application Auto Scaling API 操作来帮助保护您的 AWS 资源。

内容

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，您需要了解哪些 IAM 功能可用于 Application Auto Scaling。

可以与 Application Auto Scaling 结合使用的 IAM 功能

IAM 功能	Application Auto Scaling 支持
基于身份的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
基于资源的策略	否
ACLs	否

IAM 功能	Application Auto Scaling 支持
ABAC（策略中的标签）	部分
临时凭证	是
服务角色	是
服务关联角色	是

要全面了解 Application Auto Scaling 和其他功能如何 AWS 服务 与大多数 IAM 功能配合使用 AWS 服务 ，请在 IAM 用户指南中查看如何与 IAM 配合使用。

Application Auto Scaling 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

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。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Application Auto Scaling 基于身份的策略示例

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。

操作

支持策略操作：是

在 IAM 策略语句中，您可以从支持 IAM 的任何服务中指定任何 API 操作。对于 Application Auto Scaling，请使用以下前缀为 API 操作命名：`application-autoscaling:`。例如：`application-autoscaling:RegisterScalableTarget`、`application-autoscaling:PutScalingPolicy` 和 `application-autoscaling:DeregisterScalableTarget`。

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*"
```

有关应用程序 Auto Scaling 操作的列表，请参阅《服务授权参考》中的 App AWS lication Auto Scaling 定义的操作。

资源

支持策略资源：是

在 IAM policy 声明中，Resource 元素指定了该声明涵盖的一个或多个对象。对于 Application Auto Scaling，每个 IAM 策略声明都适用于您使用其亚马逊资源名称 (ARNs) 指定的可扩展目标。

可扩展目标的 ARN 资源格式：

```
arn:aws:application-autoscaling:region:account-id:scalable-target/unique-identifier
```

例如，您可以在语句中使用 ARN 指示特定的可扩展目标，如下所示。唯一 ID（1234abcd56ab78cd901ef1234567890ab123）是 Application Auto Scaling 分配给可扩展目标的值。

```
"Resource": "arn:aws:application-autoscaling:us-east-1:123456789012:scalable-
target/1234abcd56ab78cd901ef1234567890ab123"
```

您可以使用通配符（*）指定属于特定账户的所有实例，如下所示。

```
"Resource": "arn:aws:application-autoscaling:us-east-1:123456789012:scalable-target/*"
```

要指定所有资源，或者如果特定 API 操作不支持 ARNs，请使用通配符 (*) 作为Resource元素，如下所示。

```
"Resource": "*"
```

有关更多信息，请参阅《服务授权参考》中的 App AWS lication Auto Scaling 定义的资源类型。

条件键

支持特定于服务的策略条件键：是

您可以在控制 Application Auto Scaling 资源访问的 IAM policy 中指定条件。仅当条件为 True 时，策略语句才有效。

Application Auto Scaling 支持以下服务定义条件键，您可以在基于身份的策略中使用这些条件键来确定谁可以执行 Application Auto Scaling API 操作。

- `application-autoscaling:scalable-dimension`
- `application-autoscaling:service-namespace`

要了解可以将条件键与哪些应用程序 Auto Scaling API 操作一起使用，请参阅《服务授权参考》中的 App AWS lication Auto Scaling 定义的操作。有关使用应用程序 Auto Scaling 条件键的更多信息，请参阅AWS 应用程序 Auto Scaling 的条件键。

要查看对所有服务都可用的全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

基于资源的策略

支持基于资源的策略：否

其他 AWS 服务，例如 Amazon 简单存储服务，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 (ACLs)

支持 ACLs：否

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

结合使用 ABAC 与 Application Auto Scaling

支持 ABAC（策略中的标签）：部分支持

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。标记实体和资源是

ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

ABAC 可用于支持标签的资源，但并非所有资源都支持标签。计划操作和扩缩策略不支持标签，但可扩展目标支持标签。有关更多信息，请参阅 Application Auto Scaling 的标签支持。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证与 Application Auto Scaling 一起使用

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

服务角色

支持服务角色：是

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 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。

服务关联角色

支持服务关联角色：是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关 Application Auto Scaling 服务相关角色的信息，请参阅 Application Auto Scaling 的服务相关角色。

AWS Application Auto Scaling 的托管策略

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略： WorkSpaces 应用程序和 CloudWatch

策略名称：AWSApplicationAutoscalingAppStreamFleetPolicy

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_AppStreamFleet允许Application Auto Scaling致电亚马逊 AppStream CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

AWS 托管策略：Aurora 和 CloudWatch

策略名称：AWSApplication自动扩缩RDSCluster策略

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_RDSCluster以允许 Application Auto Scaling 调用 Aurora CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

AWS 托管策略：亚马逊 Comprehend 和 CloudWatch

策略名称：AWSApplicationAutoscalingComprehendEndpointPolicy

此策略附加AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint到名为的服务相关角色，允许应用程序 Auto Scaling 调用 Amazon CloudWatch Comprehend 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

AWS 托管策略：DynamoDB 和 CloudWatch

策略名称：AWSApplicationAutoscalingDynamoDBTable政策

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_DynamoDBTable以允许 Application Auto Scaling 调用 Dynamo DBand CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

AWS 托管策略：Amazon ECS 和 CloudWatch

策略名称：AWSApplication自动扩缩ECSService策略

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_ECSService以允许 Application Auto Scaling 调用 Amazon ECS CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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:PutMetricAlarm`
- 操作：`cloudwatch:DescribeAlarms`
- 操作：`cloudwatch:GetMetricData`
- 操作：`cloudwatch:DeleteAlarms`

AWS 托管策略： ElastiCache 和 CloudWatch

策略名称：AWSApplicationAutoscalingElastiCacheRGPolicy

此策略附加到名为的服务相关角色，允许 Appl AWSServiceRoleForApplicationAutoScaling_ElastiCacheRGication Auto Scaling 代表您调用 ElastiCache CloudWatch 和执行扩展。这个与服务相关的角色可以用于 ElastiCache Memcached、Redis OSS 和 Valkey。

权限详细信息

此权限策略支持 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 `elasticache:ModifyCacheCluster`
- 操作：资源 `arn:aws:cloudwatch:*:*:alarm:*` 上的 `cloudwatch:DescribeAlarms`
- 操作：资源 `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:PutMetricAlarm`
- 操作：资源 `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:DeleteAlarms`

AWS 托管策略：Amazon Keyspaces 和 CloudWatch

策略名称：AWSApplicationAutoscalingCassandraTablePolicy

此策略附加AWSServiceRoleForApplicationAutoScaling_CassandraTable到名为的服务相关角色，允许应用程序 Auto Scaling 调用 Amazon Keyspaces CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：对以下资源执行 `cassandra:Select`：

  - arn:*:cassandra:*:*:/keyspace/system/table/*
  - arn:*:cassandra:*:*:/keyspace/system_schema/table/*
  - arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*
- 操作：所有资源上的 cassandra:Alter
- 操作：所有资源上的 cloudwatch:DescribeAlarms
- 操作：所有资源上的 cloudwatch:PutMetricAlarm
- 操作：所有资源上的 cloudwatch:DeleteAlarms

AWS 托管策略：Lambda 和 CloudWatch

策略名称：AWSApplicationAutoscalingLambdaConcurrencyPolicy

此策略附加到名为的服务相关角色，允许 Applic AWSServiceRoleForApplicationAutoScaling_LambdaConcurrencyation Auto Scaling 代表您调用 Lambda CloudWatch 并执行扩展。

权限详细信息

此权限策略支持 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

AWS 托管策略：Amazon MSK 和 CloudWatch

策略名称：AWSApplicationAutoscalingKafkaClusterPolicy

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_KafkaCluster以允许 Application Auto Scaling 调用 Amazon MSK CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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

AWS 托管策略：Neptune 和 CloudWatch

策略名称：AWSApplicationAutoscalingNeptuneClusterPolicy

此策略附加到名为的服务相关角色，AWSServiceRoleForApplicationAutoScaling_NeptuneCluster以允许 Application Auto Scaling 调用 Neptune CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：所有资源上的 rds:ListTagsForResource
- 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：所有资源上的 cloudwatch:DescribeAlarms
- 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
- 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：资源 arn:aws:rds:*:*:db:autoscaled-reader* 上的 rds:DeleteDBInstance
- 操作：资源 arn:aws:cloudwatch:*:*:alarm:TargetTracking* 上的 cloudwatch:PutMetricAlarm

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

AWS 托管策略： SageMaker AI 和 CloudWatch

策略名称：AWSApplicationAutoscalingSageMakerEndpointPolicy

此策略附加AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint到名为的服务相关角色，允许 Application Auto Scaling 代表您调用 SageMaker AI CloudWatch 并执行扩展。

权限详细信息

此权限策略支持 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：所有资源上的 `sagemaker:DescribeEndpoint`
- 操作：所有资源上的 `sagemaker:DescribeEndpointConfig`
- 操作：所有资源上的 `sagemaker:DescribeInferenceComponent`
- 操作：所有资源上的 `sagemaker:UpdateEndpointWeightsAndCapacities`
- 操作：所有资源上的 `sagemaker:UpdateInferenceComponentRuntimeConfig`
- 操作：所有资源上的 `cloudwatch:DescribeAlarms`
- 操作：所有资源上的 `cloudwatch:GetMetricData`
- 操作：资源 `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:PutMetricAlarm`
- 操作：资源 `arn:aws:cloudwatch:*:*:alarm:TargetTracking*` 上的 `cloudwatch:DeleteAlarms`

AWS 托管策略：EC2 Spot 队列和 CloudWatch

策略名称：AWSApplication自动扩展 EC2 SpotFleetRequestPolicy

此策略附加到名为 AWSServiceRoleForApplicationAutoScaling_EC2 的服务相关角色，允许 Applic SpotFleetRequest ation Auto Scaling 调用 Amazon EC2 CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

AWS 托管策略： WorkSpaces 和 CloudWatch

策略名称：AWSApplicationAutoscalingWorkSpacesPoolPolicy

此策略附加到名为的服务相关角色，允许 Appl AWSServiceRoleForApplicationAutoScaling_WorkSpacesPoolication Auto Scaling 代表您调用 WorkSpaces CloudWatch 和执行扩展。

权限详细信息

此权限策略支持 Application Auto Scaling 对指定的资源完成以下操作：

- 操作：对与 SLR 来自同一账户的所有资源执行 `workspaces:DescribeWorkspacesPools`
- 操作：对与 SLR 来自同一账户的所有资源执行 `workspaces:UpdateWorkspacesPool`
- 操作：对与 SLR 来自同一账户的所有警报执行 `cloudwatch:DescribeAlarms`
- 操作：对与 SLR 来自同一账户的所有警报执行 `cloudwatch:PutMetricAlarm`，其中警报名称以 TargetTracking 开头
- 操作：对与 SLR 来自同一账户的所有警报执行 `cloudwatch:DeleteAlarms`，其中警报名称以 TargetTracking 开头

AWS 托管策略：自定义资源和 CloudWatch

策略名称：AWSApplicationAutoScalingCustomResourcePolicy

此策略附加AWSServiceRoleForApplicationAutoScaling_CustomResource到名为的服务相关角色，允许 Application Auto Scaling 调用通过 API Gateway 提供的自定义资源 CloudWatch 并代表您执行扩展。

权限详细信息

此权限策略支持 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`

Application Auto Scaling 更新 AWS 了托管策略

查看自该服务开始跟踪这些更改以来 Application Auto Scaling AWS 托管策略更新的详细信息。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	描述	日期
AWSApplicationAutoscalingElastiCacheRGPolicy— 更新现有政策	添加了调用 ElastiCache `ModifyCacheCluster` API 操作以支持 Memcached 自动缩放的权限。	2025 年 4 月 10 日
AWSApplication自动扩缩ECSService政策-更新现有政策	添加了调用 CloudWatch `GetMetricData` API 操作以支持预测性扩展的权限。	2024 年 11 月 21 日
AWSApplicationAutoscalingWorkSpacesPoolPolicy：新策略	为 Amazon 添加了托管策略 WorkSpaces。此策略附加到服务相关角色，该角色允许 Application Auto Scaling 代表您调用 WorkSpaces CloudWatch 和执行扩展。	2024 年 6 月 24 日
AWSApplicationAutoscalingSageMakerEndpointPolicy：对现有策略的更新	增加了调用 SageMaker A `UpdateInferenceComponentRuntimeConfig` l`DescribeInferenceComponent` 和 API 操作的权限，以支持兼容即将到来的集成 SageMaker AI 资源的自动缩放。现在，该策略还将 CloudWatch `PutMetric`	2023 年 11 月 13 日

更改	描述	日期
	`Alarm` 和 `DeleteAlarms` API 操作限制为与目标跟踪扩展策略一起使用的 CloudWatch 警报。	
AWSApplicationAutoscalingNeptuneClusterPolicy：新策略	为 Neptune 添加了托管式策略。此策略附加到服务相关角色，该角色允许 Application Auto Scaling 调用 Neptune CloudWatch 并代表您执行扩展。	2021 年 10 月 6 日
AWSApplication自动扩缩RDSCluster政策-新政策	为添加了托管策略 ElastiCache。此策略附加到服务相关角色，该角色允许 Application Auto Scaling 代表您调用 ElastiCache CloudWatch 和执行扩展。	2021 年 8 月 19 日
Application Auto Scaling 已开启跟踪更改	Application Auto Scaling 开始跟踪其 AWS 托管策略的更改。	2021 年 8 月 19 日

Application Auto Scaling 的服务相关角色

Application Auto Scaling 使用服务相关角色来获得代表您调用其他 AWS 服务所需的权限。服务相关角色是一种独特的 AWS Identity and Access Management (IAM) 角色，直接链接到 AWS 服务。服务相关角色提供了一种向服务委派权限的安全方式， AWS 因为只有关联的服务才能担任服务相关角色。

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 服务相关角色 ARN 参考。

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 AWS Application Auto Scaling 的托管策略。

内容

创建服务相关角色所需的权限

Application Auto Scaling 需要权限才能在您中的任何用户首次 AWS 账户 调用`RegisterScalableTarget`给定服务时创建服务相关角色。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"
```

以下是一个基于身份的策略，该策略授予为竞价型实例集创建服务相关角色的权限。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。

JSON

```
{
    "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 AWS 服务 可以与 Application Auto Scaling 一起使用。

创建服务相关角色（自动）

您无需手动创建服务关联角色。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 控制台或 IAM API。 AWS CLI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 (AWS CLI)

使用以下create-service-linked-role命令创建 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 AWS 服务 可以与 Application Auto Scaling 一起使用 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 AWS 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描述。

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 AWS 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，请参阅亚马逊弹性容器服务开发人员指南中的删除 Amazon ECS 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 支持在提供服务的所有 AWS 区域中使用服务相关角色。

服务相关角色 ARN 参考

下表列出了每个 AWS 服务 与 Application Auto Scaling 配合使用的服务相关角色的亚马逊资源名称 (ARN)。

服务	进行筛选
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`

服务	进行筛选
Comprehend	arn:aws:iam:: *012345678910* :role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint
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

服务	进行筛选
Neptune	arn:aws:iam:: *012345678910* :role/aws-service-role/neptune.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_NeptuneCluster
SageMaker AI	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
WorkSpaces	arn:aws:iam:: *012345678910* :role/aws-service-role/workspaces.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_WorkSpacesPool
自定义资源	arn:aws:iam:: *012345678910* :role/aws-service-role/custom-resource.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CustomResource

> **Note**
>
> 即使指定的服务相关角色尚不存在，您也可以在 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。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，您中的全新用户 AWS 账户 无权执行任何操作。IAM 管理员必须创建并分配 IAM policy，以便为 IAM 身份（例如用户或角色）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。

内容

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。编写基于身份的策略时，请参阅本节。每个策略授予执行全部或部分 Application Auto Scaling API 操作的权限。您还需要确保最终用户拥有目标服务的权限，以及 CloudWatch（有关详细信息，请参阅下一节）。

以下基于身份的策略授予执行全部 Application Auto Scaling API 操作的权限。

JSON

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

以下基于身份的策略授予执行配置扩展策略而非计划操作所需的全部 Application Auto Scaling API 操作的权限。

JSON

```
{
    "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": "*"
        }
    ]
}
```

以下基于身份的策略授予执行配置计划操作而非扩展策略所需的全部 Application Auto Scaling API 操作的权限。

JSON

```
{
    "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": "*"
        }
    ]
}
```

对目标服务进行 API 操作所需的权限以及 CloudWatch

要成功配置并将 Application Auto Scaling 与目标服务一起使用，必须向最终用户授予访问亚马逊 CloudWatch 和他们将为其配置扩展的每项目标服务的权限。使用以下策略授予使用目标服务和所需的最低权限 CloudWatch。

内容

- AppStream 2.0 支舰队
- Aurora 副本
- Amazon Comprehend 文档分类和实体识别程序终端节点
- DynamoDB 表和全局二级索引
- ECS 服务
- ElastiCache 复制组
- Amazon EMR 集群
- Amazon Keyspaces 表
- Lambda 函数
- Amazon Managed Streaming for Apache Kafka (MSK) 代理存储
- Neptune 集群
- SageMaker AI 端点
- Spot 实例集（Amazon EC2）
- 自定义资源

AppStream 2.0 支舰队

以下基于身份的策略授予所需的所有 AppStream 2.0 和 CloudWatch API 操作的权限。

JSON

```
{
```

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

Aurora 副本

以下基于身份的策略授予对所有 Aurora 和 CloudWatch API 所需操作的权限。

JSON

```
{
    "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

以下基于身份的策略向所有必需的 Amazon Com CloudWatch prehend 和 API 操作授予权限。

JSON

```
{
    "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

以下基于身份的策略向所有必需的 DynamoDB 和 API 操作授予权限。 CloudWatch

JSON

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

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

ECS 服务

以下基于身份的策略向所有必需的 ECS 和 CloudWatch API 操作授予权限。

JSON

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

ElastiCache 复制组

以下基于身份的策略向所有 ElastiCache 必需的 CloudWatch API 操作授予权限。

JSON

```
{
    "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

以下基于身份的策略向所有必需的 Amazon EMR 和 CloudWatch API 操作授予权限。

JSON

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

Amazon Keyspaces 表

以下基于身份的策略向所有 Amazon Keyspaces 和 CloudWatch API 操作授予权限。

JSON

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

Lambda 函数

以下基于身份的策略向所有必需的 Lambda 和 CloudWatch API 操作授予权限。

JSON

```
{
    "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

以下基于身份的策略向所有必需的 Amazon MSK 和 CloudWatch API 操作授予权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "kafka:DescribeCluster",
              "kafka:DescribeClusterOperation",
              "kafka:UpdateBrokerStorage",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Neptune 集群

以下基于身份的策略向所有必需的 Neptune 和 CloudWatch API 操作授予权限。

JSON

```
{
    "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 AI 端点

以下基于身份的策略向所有必需的 SageMaker A CloudWatch I 和 API 操作授予权限。

JSON

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

Spot 实例集（Amazon EC2）

以下基于身份的策略向所有必需的 Spot 队列和 CloudWatch API 操作授予权限。

JSON

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

自定义资源

以下基于身份的策略授予执行 API Gateway API 操作的权限。该策略还授予 CloudWatch 执行所有必需操作的权限。

JSON

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

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

在中工作的权限 AWS 管理控制台

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，每项服务都提供 AWS 托管（预定义）IAM 策略，这些策略定义了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 AWS 管理控制台中查看和处理特定 Application Auto Scaling API 操作的精细权限。您可以使用前面部分中的示例策略；但是，它们是为使用 AWS CLI 或 SDK 发出的请求而设计的。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，要配置分步缩放，用户可能需要额外的权限才能创建和管理 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 AWS CloudTrail等服务。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。

以下基于身份的策略授予为竞价型实例集配置扩展策略的权限。除了竞价型实例集的 IAM 权限之外，从 Amazon EC2 控制台访问实例集扩展设置的控制台用户必须具有使用支持动态扩展的服务的适当权限。

JSON

```
{
    "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"
                }
            }
        }
    ]
}
```

该策略允许控制台用户在 Amazon EC2 控制台中查看和修改扩展策略，并在控制 CloudWatch 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。

您也可以根据需要调整 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 CloudWatch 控制台所需权限。

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`AccessDeniedException`在调用 AWS API 操作时收到，则表示您正在使用的 AWS Identity and Access Management (IAM) 证书没有进行该调用所需的权限。

如果 `mateojackson` 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。

您的账户管理员需要确保您拥有访问所有 API 操作的权限，Application Auto Scaling 使用这些操作来访问目标服务中的资源和 CloudWatch。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，我的 IAM policy 返回错误或未按预期工作

除了 Application Auto Scaling 操作外，您的 IAM 策略还必须授予调用目标服务的权限和 CloudWatch。如果用户或应用程序没有这些额外的权限，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 IAM policy，请参阅 Application Auto Scaling 基于身份的策略示例 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 Application Auto Scaling API 调用的权限验证。

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。

对目标资源进行 Application Auto Scaling API 调用的权限验证

向 Application Auto Scaling API 操作发出授权请求要求 API 调用者必须具有访问目标服务中和中的 AWS 资源的权限 CloudWatch。在继续处理请求 CloudWatch 之前，Application Auto Scaling 会验证与目标服务关联的请求的权限。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。

作为其工作原理的示例，以下信息提供了有关应用程序 Auto Scaling 如何使用 Aurora 和 CloudWatch 执行权限验证的详细信息。

当用户针对 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用户是否具有所需的权限（以粗体显示）。

- RDS: Cre ateDBInstance：为了确定用户是否拥有此权限，我们向 `CreateDBInstance` API 操作发送请求，尝试在用户指定的 Aurora 数据库集群中创建参数无效（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: Delete DBInstance：我们向 API 操作发送了一个空的`DeleteDBInstance`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）。
- rdsAddTagsToResource:: 由于 `AddTagsToResource` API 操作需要亚马逊资源名称 (ARN)，因此必须使用无效的账户 ID (12345) 和虚拟实例 ID () 指定 “虚拟” 资源non-existing-db来构建 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: Des cribeDBClusters：我们描述了为自动缩放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。
- RDSDBInstances: Des cribe：我们使用`db-cluster-id`过滤器调用 `DescribeDBInstances` API，该过滤器过滤用户为注册可扩展目标而提供的集群名称。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。
- cloudwatchPutMetricAlarm:: 我们在调用 `PutMetricAlarm` API 时不带任何参数。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。
- cloudwatchDescribeAlarms:: 我们在调用 `DescribeAlarms` API 时将最大记录数值设置为 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。

- cloudwatchDeleteAlarms:: 与PutMetricAlarm上述类似，我们不提供任何参数可供DeleteAlarms请求。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。您可以使用采取措施手动识别哪些呼叫未通过验证 AWS CloudTrail。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。

> **Note**
>
> 如果您使用收到有关应用程序 Auto Scaling 事件的警报 CloudTrail，则默认情况下，这些警报将包括用于验证用户权限的应用程序 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。

使用接口 VPC 端点访问 Application Auto Scaling

您可以使用 AWS PrivateLink 在您的 VPC 和 Application Auto Scaling 之间创建私有连接。您可以像在 VPC 中一样访问 Application Auto Scaling，无需使用互联网网关、NAT 设备、VPN Direct Connect 连接或连接。VPC 中的实例不需要公有 IP 地址即可访问 Application Auto Scaling。

您可以通过创建由 AWS PrivateLink提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管的网络接口，用作发往 Application Auto Scaling 的流量的入口点。

有关更多信息，请参阅AWS PrivateLink 指南 AWS PrivateLink中的AWS 服务 通过访问。

内容

创建接口 VPC 端点

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AWS PrivateLink 指南中的使用接口 VPC 终端节点访问 AWS 服务。

您不需要更改任何其他设置。Application Auto Scaling 使用 AWS 服务终端节点或私有接口 VPC 终端节点调用其他服务，以使用中者为准。

创建 VPC 端点策略

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：

- 可执行操作的主体。
- 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《AWS PrivateLink 指南》中的 VPC 终端节点策略。

Application Auto Scaling 中的恢复功能

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。

AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。

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失效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、容错能力和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

Application Auto Scaling 中的基础设施安全性

作为一项托管服务，Application Auto Scaling 受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Application Auto Scaling。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

Application Auto Scaling 的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

Application Auto Scaling 配额

您的每个配额 AWS 账户 都有默认配额，以前称为限制 AWS 服务。除非另有说明，否则，每个限额是区域特定的。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 AWS 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。

您 AWS 账户 有以下与 Application Auto Scaling 相关的配额。

名称	默认值	可调整
每种资源类型的可扩展目标数	Amazon DynamoDB：5000 | Amazon ECS：3000 | Amazon Keyspaces：1500 | 其它资源类型：500	是
每个可扩展目标的扩展策略数（包括步进扩展和目标跟踪策略）	50	不可以
每个可扩展目标的计划操作	200	否
每个步进扩展策略的步进调整	20	是

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。有关其他亚马逊云科技默认配额的信息，请参阅 Amazon Web Services 一般参考 中的服务端点和配额。

Application Auto Scaling 的文档历史记录

下表介绍了自 2018 年 1 月以来对 Application Auto Scaling 文档的重要补充。如需有关此文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
添加对 ElastiCache Memcached 集群的支持	使用 Application Auto Scaling 水平扩展 Memcached 集群的节点数量。有关更多信息，请参阅ElastiCache 和 Application Auto Scaling。	2025 年 4 月 10 日
AWS 托管策略更新	Application Auto Scaling 更新了`AWSApplicationAutoscalingElastiCacheRGPolicy` 政策。	2025 年 4 月 10 日
指南更改	《App lication Auto Scaling 用户指南》中的新主题可帮助您开始在 Application Auto Scaling 中使用预测性扩展。请参阅 App lication Auto Scaling 预测扩展。	2024 年 11 月 21 日
AWS 托管策略更新	Application Auto Scaling 更新了`AWSApplicationAutoscalingECSServicePolicy` 政策。	2024 年 11 月 21 日
添加对资源池的支持 WorkSpaces	使用 Application Auto Scaling 来扩展池 WorkSpaces。有关更多信息，请参阅 Amazon WorkSpaces 和 Application Auto Scaling。“App lication Auto Scaling 更新 AWS 托管策	2024 年 6 月 27 日

	略”主题已更新，列出了与集成的新托管策略 WorkSpaces。	
指南更改	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。	2024 年 1 月 16 日
支持 A SageMaker I 推理组件	使用 Application Auto Scaling 扩展推理组件的副本。	2023 年 11 月 29 日
AWS 托管策略更新	Application Auto Scaling 更新了`AWSApplicationAutoscalingSageMakerEndpointPolicy` 政策。	2023 年 11 月 13 日
支持 SageMaker AI 无服务器配置的并发性	使用 Application Auto Scaling 扩展无服务器端点的预置并发。	2023 年 5 月 9 日
使用标签对您的可扩展目标进行分类	您可以将自己的元数据以标签的形式分配给 Application Auto Scaling 可扩展目标。请参阅 Application Auto Scaling 标签支持。	2023 年 3 月 20 日
Support 支持 CloudWatch 公制数学	创建目标跟踪扩展策略时，您现在可使用指标数学。使用指标数学，您可以查询多个 CloudWatch 指标，并使用数学表达式根据这些指标创建新的时间序列。请参阅使用指标数学为 Application Auto Scaling 创建目标跟踪扩展策略	2023 年 3 月 14 日

不扩展的原因	现在，您可以使用 Application Auto Scaling API，检索 Application Auto Scaling 不扩展资源的机器可读原因。请参阅 Scaling activities for Application Auto Scaling（Application Auto Scaling 的扩展活动）。	2023 年 1 月 4 日
指南更改	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	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 AWS 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
Application Auto Scaling 现在会报告其 AWS 托管策略的更改	从 2021 年 8 月 19 日起，托管策略的更改将在主题 “Application Auto Scaling 对 AWS 托管策略的更新” 中报告。列出的第一个更改是添加了 ElastiCache (Redis OSS) 所需的权限。	2021 年 8 月 19 日

添加对 ElastiCache (Redis OSS) 复制组的支持	使用 Application Auto Scaling 来扩展 ElastiCache （Redis OSS）复制组（集群）的节点组数量和每个节点组的副本数量。有关更多信息，请参阅 ElastiCache (Redis OSS) 和 Application Auto Scaling。	2021 年 8 月 19 日
指南更改	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。还为目标服务和亚马逊上的操作添加了新的示例 IAM 权限策略 CloudWatch。有关更多信息，请参阅使用 AWS CLI 或 SDK 的策略示例。	2021 年 2 月 23 日
添加对本地时区的支持	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	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。	2020 年 10 月 15 日
添加对 Amazon Managed Streaming for Apache Kafka 集群存储的支持	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	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	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	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	各种改进和更正。	2020 年 1 月 15 日
增加了通知功能	Application Auto Scaling 现在会向亚马逊发送事件，EventBridge 并在发生某些操作 AWS Health Dashboard 时向您发送通知。有关更多信息，请参阅 Application Auto Scaling 监控。	2019 年 12 月 20 日
添加对 AWS Lambda 函数的支持	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	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 WorkSpaces 应用程序对目标跟踪扩展策略的支持	使用目标跟踪扩展策略来扩展 WorkSpaces 应用程序队列的大小。	2019 年 11 月 25 日
对 Amazon VPC 端点的支持	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	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
指南更改	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	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 AI 端点变体的支持	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	描述	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	2017 年 11 月 17 日

更改	描述	日期
	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。	
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 7 月 12 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 WorkSpaces 应用程序队列的支持	使用 Application Auto Scaling 扩展队列的规模。有关更多信息，请参阅《亚马逊 WorkSpaces 应用程序管理指南》中的 Fleet Auto Scaling WorkSpaces 应用程序管理指南。	2017 年 3 月 23 日

更改	描述	日期
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《Amazon EC2 用户指南》中的竞价型实例集的自动扩展。	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

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。