Application Auto Scaling

User Guide

Application Auto Scaling: User Guide

Table of Contents

What is Application Auto Scaling?

Application Auto Scaling is a web service for developers and system administrators who need a solution for automatically scaling their scalable resources for individual AWS services beyond Amazon EC2. Application Auto Scaling allows you to configure automatic scaling for the following resources:

- AppStream 2.0 fleets
- Aurora replicas
- Amazon Comprehend document classification and entity recognizer endpoints
- DynamoDB tables and global secondary indexes
- Amazon Elastic Container Service (ECS) services
- ElastiCache for Redis clusters (replication groups)
- Amazon EMR clusters
- Amazon Keyspaces (for Apache Cassandra) tables
- Lambda function provisioned concurrency
- Amazon Managed Streaming for Apache Kafka (MSK) broker storage
- Amazon Neptune clusters
- SageMaker endpoint variants
- Spot Fleet requests
- Custom resources provided by your own applications or services. For more information, see the GitHub repository.

To see the regional availability for any of the AWS services listed above, see the Region table.

For information about scaling your fleet of Amazon EC2 instances using Auto Scaling groups, see the Amazon EC2 Auto Scaling User Guide.

You can also use AWS Auto Scaling to create scaling plans to scale resources across multiple services. For more information, see the AWS Auto Scaling User Guide.

Features of Application Auto Scaling

Application Auto Scaling allows you to automatically scale your scalable resources according to conditions that you define.

- **Target tracking scaling**—Scale a resource based on a target value for a specific CloudWatch metric.
- **Step scaling**— Scale a resource based on a set of scaling adjustments that vary based on the size of the alarm breach.
- **Scheduled scaling**—Scale a resource based on the date and time.

Accessing Application Auto Scaling

If you've signed up for an AWS account, access Application Auto Scaling by signing into the AWS Management Console. Then, open the service console for one of the resources listed in the introduction. Ensure that you open the console in the same AWS Region as the resource that you want to work with.

Note
Console access is not available for all resources. For more information, see AWS services that you can use with Application Auto Scaling (p. 15).

Query API

You can also access Application Auto Scaling using the Application Auto Scaling API. Application Auto Scaling provides a Query API. These requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named `Action`. For more information, see Actions in the *Application Auto Scaling API Reference*.

AWS SDKs

If you prefer to build applications using language-specific APIs instead of submitting a request over HTTP or HTTPS, AWS provides libraries, sample code, tutorials, and other resources for software developers. These libraries provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses, making it is easier for you to get started. For more information, see AWS SDKs and tools.

Command line interface

If you prefer to use a command line interface, you have the following options:

AWS Command Line Interface (AWS CLI)

Provides commands for a broad set of AWS products, and is supported on Windows, macOS, and Linux. To get started, see AWS Command Line Interface User Guide. For more information, see application-autoscaling in the *AWS CLI Command Reference*.

AWS Tools for Windows PowerShell

Provides commands for a broad set of AWS products for those who script in the PowerShell environment. To get started, see the AWS Tools for Windows PowerShell User Guide. For more information, see the AWS Tools for PowerShell Cmdlet Reference.

CloudFormation

Application Auto Scaling is also supported in AWS CloudFormation. For more information, see Creating Application Auto Scaling resources with AWS CloudFormation (p. 108).

Setting up

Before using Application Auto Scaling to configure automatic scaling, create an AWS account, configure access permissions, and set up the AWS Command Line Interface (AWS CLI).

Topics

Sign up for an AWS account

When you sign up for an account with Amazon Web Services, your account is automatically signed up for all Amazon Web Services, including Application Auto Scaling. You are charged only for the services that you use.

If you don't already have an AWS account, you need to create one. If you already have an AWS account, you can skip the steps for creating an account in the following procedure and move to creating an IAM user in step 3.

To sign up for an AWS account

1. Open https://aws.amazon.com/ and choose **Sign Up**.
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad. AWS sends you a confirmation email after the sign-up process is complete.
3. Create an AWS Identity and Access Management (IAM) admin user. See Creating your first IAM user and group in the *IAM User Guide* for instructions.

   Important
   The getting started exercises in this guide assume that you have a user (`adminuser`) with administrator permissions. Follow the procedure to create `adminuser` in your account.
4. Make sure that you have an access key ID and a secret access key associated with the IAM user that you just created. For more information, see Access key and secret access key in the *AWS Command Line Interface User Guide*.

For more information about IAM, see the following:

- AWS Identity and Access Management (IAM)
- Getting started
- IAM User Guide

Using Application Auto Scaling in AWS Regions

Application Auto Scaling is available in multiple AWS Regions. For a list of available Regions, see the Regions and endpoints table in the *AWS General Reference*. A global AWS account allows you to work with resources in most Regions. When using Application Auto Scaling with resources in the China Regions, keep in mind that you must have a separate Amazon Web Services (China) account. In addition, there are some differences in how Application Auto Scaling is implemented. For more information on using Application Auto Scaling in the China Regions, see Application Auto Scaling in China.

Set up the AWS CLI

The AWS Command Line Interface (AWS CLI) is a unified developer tool for managing AWS services, including Application Auto Scaling. Follow the steps to download and configure the AWS CLI.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:
   - Installing the AWS CLI
   - Configuring the AWS CLI
2. Run the following command to verify that the Application Auto Scaling commands for the AWS CLI are installed.

```
aws application-autoscaling help
```

3. Add a named profile for the administrator user in the AWS CLI config file. You can use this profile when executing AWS CLI commands. For more information about named profiles, see Named profiles in the *AWS Command Line Interface User Guide*.

```
aws configure --profile adminuser
```

   When prompted, specify the AWS access key and secret access key of the IAM user to use with Application Auto Scaling.

```
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
default output format = json
```

   For a list of available AWS Regions, see Application Auto Scaling regions and endpoints in the *Amazon Web Services General Reference*.
4. To confirm that the AWS CLI profile is configured correctly, run the following command in a command window.

```
aws configure --profile adminuser
```

   If your profile has been configured correctly, you should see output similar to the following.

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

After you set up an AWS account and the AWS CLI, you can try the next tutorial, in which you configure sample scheduled scaling actions.

Getting started using the AWS CLI

In this tutorial, you use the AWS CLI to explore Application Auto Scaling. Before you begin, make sure that you have an AWS account and that you've set up the AWS CLI. For more information, see Setting up (p. 3). In this tutorial, you create scheduled actions to scale your scalable resources based on a schedule. With scheduled scaling, you can specify either a one-time action or a recurring action.

The exercises in this tutorial assume that you are using administrator credentials (`adminuser` profile) that you set up in Set up the AWS CLI (p. 4). If you don't provide this profile, the default profile is assumed. Note that to create, update, delete, or list Application Auto Scaling resources, you need permissions to perform the action, and you need permission to access the corresponding resources. For more information, see Identity and Access Management for Application Auto Scaling (p. 79).

When using the AWS CLI, remember that your commands run in the AWS Region that's configured for your profile. If you want to run the commands in a different Region, either change the default Region for your profile, or use the `--region` parameter with the command.

Note
You may incur AWS charges as part of this tutorial. Please monitor your Free tier usage and make sure that you understand the costs associated with the number of units of read and write capacity that your DynamoDB databases uses.

Contents

Step 1: Register your scalable target

Begin by registering your resource as a scalable target with Application Auto Scaling. A scalable target is a resource that Application Auto Scaling can scale out or scale in.

You can use any resource that works with Application Auto Scaling and supports scheduled scaling, but for these examples, let's assume that you want to scale a DynamoDB table called `my-table`. If you don't already have a DynamoDB table, you can create one now (Step 1: Create a DynamoDB table in the *Amazon DynamoDB Developer Guide*).

To use a DynamoDB global secondary index or a resource for a different service, update the examples accordingly. Specify its namespace in `--service-namespace`, its scalable dimension in `--scalable-dimension`, and its resource ID in `--resource-id`. For a list of valid values for each option, see register-scalable-target.

To register your scalable target with Application Auto Scaling

1. (Optional) Use the describe-scalable-targets command to check whether any DynamoDB resources are already registered. This helps you verify whether to register the `my-table` table. For example, if you previously configured automatic scaling for this table from the DynamoDB console, it may already be registered with Application Auto Scaling.

   Linux, macOS, or Unix

   ```
   aws application-autoscaling describe-scalable-targets \
   ```

```
  --service-namespace dynamodb \
  --profile adminuser
```

Windows

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

If there are no existing scalable targets, this is the response.

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

2. Use the following register-scalable-target command to register or update the write capacity of a DynamoDB table called `my-table`. Set a minimum desired capacity of 5 write capacity units and a maximum desired capacity of 10 write capacity units.

   Linux, macOS, or Unix

```
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 \
  --profile adminuser
```

   Windows

```
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 --profile adminuser
```

   This command does not return any output if it is successful.

Step 2: Create two scheduled actions

Application Auto Scaling allows you to schedule the time when a scaling action should occur. You specify the scalable target, the schedule, and the minimum and maximum capacity. At the specified time, Application Auto Scaling updates the minimum and maximum value for the scalable target. If its current capacity is outside of this range, this results in a scaling activity.

Scheduling updates to the minimum and maximum capacity is also helpful if you decide to create a scaling policy. A scaling policy allows your resources to scale dynamically based on current resource utilization. A common guardrail for a scaling policy is having appropriate values for minimum and maximum capacity.

For this exercise, we create two one-time actions for scale out and scale in.

To create and view the scheduled actions

1. To create the first scheduled action, use the following put-scheduled-action command.

   The **at** command in `--schedule` schedules the action to be run once at a specified date and time in the future. Hours are in 24-hour format in UTC. Schedule the action to occur about 5 minutes from now.

At the date and time specified, Application Auto Scaling updates the `MinCapacity` and `MaxCapacity` values. Assuming the table currently has 5 write capacity units, Application Auto Scaling scales out to `MinCapacity` to put the table within the new desired range of 15-20 write capacity units.

Linux, macOS, or Unix

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

Windows

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

This command does not return any output if it is successful.

2. To create the second scheduled action that Application Auto Scaling uses to scale in, use the following put-scheduled-action command.

   Schedule the action to occur about 10 minutes from now.

   At the date and time specified, Application Auto Scaling updates the table's `MinCapacity` and `MaxCapacity`, and scales in to `MaxCapacity` to return the table to the original desired range of 5-10 write capacity units.

   Linux, macOS, or Unix

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

   Windows

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

3. (Optional) Get a list of scheduled actions for the specified service namespace using the following describe-scheduled-actions command.

   Linux, macOS, or Unix

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

Windows

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

The following is example output.

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:123456789012: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-
east-1:123456789012: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"
        }
    ]
}
```

Step 3: View the scaling activities

In this step, you view the scaling activities triggered by the scheduled actions, and then verify that DynamoDB changed the table's write capacity.

To view the scaling activities

1. Wait for the time you chose, and verify that your scheduled actions are working by using the following describe-scaling-activities command.

   Linux, macOS, or Unix

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

Windows

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

The following is example output for the first scheduled action while the scheduled action is in progress.

Scaling activities are ordered by creation date, with the newest scaling activities returned first.

```
{
    "ScalingActivities": [
        {
            "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",
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Waiting for
 change to be fulfilled by dynamodb.",
            "StatusCode": "InProgress"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/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"
        }
    ]
}
```

The following is example output after both scheduled actions have run.

```
{
    "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"
        }
    ]
}
```

2. After running the scheduled actions successfully, go to the DynamoDB console and choose the table that you want to work with. View the **Write capacity units** under the **Capacity** tab. After the second scaling action ran, the write capacity units should have been scaled from 15 to 10.

   You can also view this information through the AWS CLI.

   Verify the table's current write capacity by using the DynamoDB describe-table command. Include the `--query` option to filter the output. For more information about the output filtering capabilities of the AWS CLI, see Controlling command output from the AWS CLI in the *AWS Command Line Interface User Guide*.

   Linux, macOS, or Unix

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

   Windows

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

   The following is example output.

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

Step 4: Next steps

Now that you have familiarized yourself with Application Auto Scaling and some of its features, consider doing the following:

- If you want to try scaling on a recurring schedule, see the tutorial in Tutorial: Configuring auto scaling to handle a heavy workload (p. 60).
- If you want to try scaling dynamically in response to changes in resource utilization (for example, by using the `DynamoDBWriteCapacityUtilization` metric), follow the steps in Target tracking scaling policies for Application Auto Scaling (p. 33).

Step 5: Clean up

When you are done working with the getting started exercises, you can clean up the associated resources as follows.

To delete the scheduled actions

The following delete-scheduled-action command deletes a specified scheduled action. You can skip this step if you want to keep the scheduled action for future use.

Linux, macOS, or Unix

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

Windows

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

To deregister the scalable target

Use the following deregister-scalable-target command to deregister the scalable target. If you have any scaling policies that you created or any scheduled actions that have not yet been deleted, they are deleted by this command. You can skip this step if you want to keep the scalable target registered for future use.

Linux, macOS, or Unix

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

Windows

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

To delete the DynamoDB table

Use the following delete-table command to delete the table that you used in this tutorial. You can skip this step if you want to keep the table for future use.

Linux, macOS, or Unix

```
aws dynamodb delete-table --table-name my-table \
  --profile adminuser
```

Windows

```
aws dynamodb delete-table --table-name my-table --profile adminuser
```

Getting started with Application Auto Scaling

This topic explains key concepts to help you learn about Application Auto Scaling and start using it.

Scalable target

An entity that you create to specify the resource that you want to scale. Each scalable target is uniquely identified by a service namespace, resource ID, and scalable dimension, which represents some capacity dimension of the underlying service. For example, an Amazon ECS service supports auto scaling of its task count, a DynamoDB table supports auto scaling of the read and write capacity of the table and its global secondary indexes, and an Aurora cluster supports scaling of its replica count.

> **Tip**
> Each scalable target also has a minimum and maximum capacity. Scaling policies will never go higher or lower than the minimum-maximum range. You can make out-of-band changes directly to the underlying resource that are outside of this range, which Application Auto Scaling doesn't know about. However, anytime a scaling policy is invoked or the `RegisterScalableTarget` API is called, Application Auto Scaling retrieves the current capacity and compares it to the minimum and maximum capacity. If it falls outside of the minimum-maximum range, then the capacity is updated to comply with the set minimum and maximum.

Scale in

When Application Auto Scaling automatically decreases capacity for a scalable target, the scalable target *scales in*. The minimum capacity of the scalable target is the lowest capacity a scaling policy can scale in to.

Scale out

When Application Auto Scaling automatically increases capacity for a scalable target, the scalable target *scales out*. The maximum capacity of the scalable target is the highest capacity a scaling policy can scale out to.

Scaling policy

A scaling policy instructs Application Auto Scaling to track a specific CloudWatch metric. Then, it determines what scaling action to take when the metric is higher or lower than a certain threshold value. For example, you might want to scale out if the CPU usage across your cluster starts to rise, and scale in when it drops again.

The metrics that are used for auto scaling are published by the target service, but you can also publish your own metric to CloudWatch and then use it with a scaling policy.

A cooldown period between scaling activities lets the resource stabilize before another scaling activity starts. Application Auto Scaling continues to evaluate metrics during the cooldown period. When the cooldown period ends, the scaling policy initiates another scaling activity if needed. While a cooldown period is in effect, if a larger scale out is necessary based on the current metric value, the scaling policy scales out immediately.

Scheduled action

Scheduled actions automatically scale resources at a specific date and time. They work by modifying the minimum and maximum capacity for a scalable target, and therefore can be used to scale in and out on a schedule by setting the minimum capacity high or the maximum capacity low. For example, you can use scheduled actions to scale an application that doesn't consume resources on weekends by decreasing capacity on Friday and increasing capacity on the following Monday.

You can also use scheduled actions to optimize the minimum and maximum values over time to adapt to situations where higher than normal traffic is expected, for example, marketing campaigns or seasonal fluctuations. Doing this can help you improve performance for times when you need to scale out higher for the increasing usage, and reduce costs at times when you use fewer resources.

Learn more

AWS services that you can use with Application Auto Scaling (p. 15) — This section introduces you to the services that you can scale and helps you set up auto scaling by registering a scalable target. It also describes each of the IAM service-linked roles that Application Auto Scaling creates to access resources in the target service.

Target tracking scaling policies for Application Auto Scaling (p. 33) — One of the primary features of Application Auto Scaling is target tracking scaling policies. Learn how target tracking policies automatically adjust desired capacity to keep utilization at a constant level based on your configured metric and target values. For example, you can configure target tracking to keep the average CPU utilization for your Spot Fleet at 50 percent. Application Auto Scaling then launches or terminates EC2 instances as required to keep the aggregated CPU utilization across all servers at 50 percent.

AWS services that you can use with Application Auto Scaling

Application Auto Scaling integrates with other AWS services so that you can add scaling capabilities to meet your application's demand. Auto scaling is an optional feature of the service that is disabled by default in almost all cases.

The following table lists the AWS services that you can use with Application Auto Scaling, including information about supported methods for configuring auto scaling. You can also use Application Auto Scaling with custom resources.

Console access – You can configure a compatible AWS service to start auto scaling by configuring a scaling policy in the console of the target service. Currently, only ElastiCache and Spot Fleet provide console support for scheduled scaling. If a service supports console access, see the *Learn more* link to learn how to configure auto scaling from that service's console.

CLI access – You can configure a compatible AWS service to start auto scaling using the AWS CLI.

SDK access – You can configure a compatible AWS service to start auto scaling using the AWS SDKs.

CloudFormation access – You can configure a compatible AWS service to start auto scaling using an AWS CloudFormation stack template. For more information, see Creating Application Auto Scaling resources with AWS CloudFormation (p. 108).

AWS service	Console access	CLI access	SDK access	CloudFormation access
AppStream 2.0 (p. 16)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Aurora (p. 17)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Amazon Comprehend (p. 18)	✗No	✓Yes	✓Yes	✓Yes
Amazon DynamoDB (p. 20)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Amazon ECS (p. 21)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Amazon ElastiCache (p. 22)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Amazon EMR	✓Yes Learn more	✓Yes	✓Yes	✓Yes

AWS service	Console access	CLI access	SDK access	CloudFormation access
Amazon Keyspaces (p. 24)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Lambda (p. 25)	✗No	✓Yes	✓Yes	✓Yes
Amazon MSK (p. 26)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Amazon Neptune (p. 27)	✗No	✓Yes	✓Yes	✓Yes
SageMaker (p. 29)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Spot Fleet (p. 30)	✓Yes Learn more	✓Yes	✓Yes	✓Yes
Custom resources (p. 31)	✗No	✓Yes	✓Yes	✓Yes

Amazon AppStream 2.0 and Application Auto Scaling

You can scale AppStream 2.0 fleets using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate AppStream 2.0 with Application Auto Scaling.

If you are just getting started with scaling AppStream 2.0 fleets, you can view sample configurations and details about using AppStream 2.0 with Application Auto Scaling in the following documentation:

- Fleet Auto Scaling for AppStream 2.0 in the *Amazon AppStream 2.0 Administration Guide*

Service-linked role created for AppStream 2.0

The following service-linked role is automatically created in your AWS account when registering AppStream 2.0 resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering AppStream 2.0 fleets as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an AppStream 2.0 fleet. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the AppStream 2.0 console, then AppStream 2.0 automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an AppStream 2.0 fleet. The following example registers the desired capacity of a fleet called `sample-fleet`, with a minimum capacity of one fleet instance and a maximum capacity of five fleet instances.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon Aurora and Application Auto Scaling

You can scale Aurora DB clusters using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate Aurora with Application Auto Scaling.

If you are just getting started with scaling Aurora DB clusters, you can view sample configurations and details about using Aurora with Application Auto Scaling in the following documentation:

- Using Amazon Aurora Auto Scaling with Aurora replicas in the *Amazon RDS User Guide*

Service-linked role created for Aurora

The following service-linked role is automatically created in your AWS account when registering Aurora resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_RDSCluster`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Aurora DB clusters as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an Aurora cluster. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the Aurora console, then Aurora automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an Aurora cluster. The following example registers the count of Aurora Replicas in a cluster called `my-db-cluster`, with a minimum capacity of one Aurora Replica and a maximum capacity of eight Aurora Replicas.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon Comprehend and Application Auto Scaling

You can scale Amazon Comprehend document classification and entity recognizer endpoints using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate Amazon Comprehend with Application Auto Scaling.

If you are just getting started with scaling Amazon Comprehend document classification and entity recognizer endpoints, you can view sample configurations and details about using Amazon Comprehend with Application Auto Scaling in the following documentation:

- Auto scaling with endpoints in the *Amazon Comprehend Developer Guide*

Service-linked role created for Amazon Comprehend

The following service-linked role is automatically created in your AWS account when registering Amazon Comprehend resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Amazon Comprehend resources as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an Amazon Comprehend document classification or entity recognizer endpoint. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

To configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a document classification endpoint. The following example registers the desired number of inference units to be used by the model for a document classifier endpoint using the endpoint's ARN, with a minimum capacity of one inference unit and a maximum capacity of three inference units.

```
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
```

  Call the `register-scalable-target` command for an entity recognizer endpoint. The following example registers the desired number of inference units to be used by the model for an entity recognizer using the endpoint's ARN, with a minimum capacity of one inference unit and a maximum capacity of three inference units.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon DynamoDB and Application Auto Scaling

You can scale DynamoDB tables and global secondary indexes using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate DynamoDB with Application Auto Scaling.

If you are just getting started with scaling DynamoDB tables and global secondary indexes, you can view sample configurations and details about using DynamoDB with Application Auto Scaling in the following documentation:

- Managing throughput capacity with DynamoDB Auto Scaling in the *Amazon DynamoDB Developer Guide*

> **Tip**
> We also provide a tutorial for scheduled scaling in Getting started using the AWS CLI (p. 5). In this tutorial, you learn the basic steps to configure scaling so your DynamoDB table scales at scheduled times.

Service-linked role created for DynamoDB

The following service-linked role is automatically created in your AWS account when registering DynamoDB resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering DynamoDB resources as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for a DynamoDB table or global secondary index. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the DynamoDB console, then DynamoDB automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a table. The following example registers the provisioned write capacity of a table called `my-table`, with a minimum capacity of five write capacity units and a maximum capacity of 10 write capacity units.

```
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
```

  Call the register-scalable-target command for a global secondary index. The following example registers the provisioned write capacity of a global secondary index called `my-table-index`, with a minimum capacity of five write capacity units and a maximum capacity of 10 write capacity units.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon ECS and Application Auto Scaling

You can scale ECS services using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate Amazon ECS with Application Auto Scaling.

If you are just getting started with scaling ECS services, you can view sample configurations and details about using Amazon ECS with Application Auto Scaling in the following documentation:

- Service Auto Scaling in the *Amazon Elastic Container Service Developer Guide*

Service-linked role created for Amazon ECS

The following service-linked role is automatically created in your AWS account when registering Amazon ECS resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_ECSService`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering ECS services as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an Amazon ECS service. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the Amazon ECS console, then Amazon ECS automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an Amazon ECS service. The following example registers a scalable target for a service called `sample-app-service`, running on the `default` cluster, with a minimum task count of one task and a maximum task count of 10 tasks.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

ElastiCache for Redis and Application Auto Scaling

You can scale ElastiCache for Redis replication groups using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate ElastiCache with Application Auto Scaling.

If you are just getting started with scaling ElastiCache for Redis replication groups, you can view sample configurations and details about using ElastiCache with Application Auto Scaling in the following documentation:

- Auto Scaling ElastiCache for Redis clusters in the *Amazon ElastiCache for Redis User Guide*

Service-linked role created for ElastiCache

The following service-linked role is automatically created in your AWS account when registering ElastiCache resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering ElastiCache for Redis replication groups as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an ElastiCache replication group. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the ElastiCache console, then ElastiCache automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an ElastiCache replication group. The following example registers the desired number of node groups for a replication group called `mycluster`, with a minimum capacity of one and a maximum capacity of five.

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

  The following example registers the desired number of replicas per node group for a replication group called `mycluster`, with a minimum capacity of 1 and a maximum capacity of 5.

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

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon Keyspaces (for Apache Cassandra) and Application Auto Scaling

You can scale Amazon Keyspaces tables using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate Amazon Keyspaces with Application Auto Scaling.

If you are just getting started with scaling Amazon Keyspaces tables, you can view sample configurations and details about using Amazon Keyspaces with Application Auto Scaling in the following documentation:

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

Service-linked role created for Amazon Keyspaces

The following service-linked role is automatically created in your AWS account when registering Amazon Keyspaces resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_CassandraTable`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Amazon Keyspaces tables as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an Amazon Keyspaces table. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the Amazon Keyspaces console, then Amazon Keyspaces automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an Amazon Keyspaces table. The following example registers the provisioned write capacity of a table called `mytable`, with a minimum capacity of five write capacity units and a maximum capacity of 10 write capacity units.

  ```
  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
  ```

  The following example registers the provisioned read capacity of a table called `mytable`, with a minimum capacity of five read capacity units and a maximum capacity of 10 read capacity units.

  ```
  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
  ```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

AWS Lambda and Application Auto Scaling

You can scale Lambda functions using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate Lambda with Application Auto Scaling.

If you are just getting started with scaling Lambda functions, you can view sample configurations and details about using Lambda with Application Auto Scaling in the following documentation:

- Managing concurrency for a Lambda function in the *AWS Lambda Developer Guide*

Service-linked role created for Lambda

The following service-linked role is automatically created in your AWS account when registering Lambda resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Lambda functions as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for a Lambda function. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

To configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a Lambda function. The following example registers the provisioned concurrency for an alias called `BLUE` for a function called `my-function`, with a minimum capacity of 0 and a maximum capacity of 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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

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

You can scale out Amazon MSK cluster storage using target tracking scaling policies. Scale in by the target tracking policy is disabled.

Use the following information to help you integrate Amazon MSK with Application Auto Scaling.

If you are just getting started with scaling Amazon MSK cluster storage, you can view details about using Amazon MSK with Application Auto Scaling in the following documentation:

- Auto-expanding storage for an Amazon MSK cluster in the *Amazon Managed Streaming for Apache Kafka Developer Guide*

Service-linked role created for Amazon MSK

The following service-linked role is automatically created in your AWS account when registering Amazon MSK resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_KafkaCluster`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Amazon MSK cluster storage as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create a scaling policy for the storage volume size per broker of an Amazon MSK cluster. A scalable target is a resource that Application Auto Scaling can scale. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the Amazon MSK console, then Amazon MSK automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an Amazon MSK cluster. The following example registers the storage volume size per broker of an Amazon MSK cluster, with a minimum capacity of 100 GiB and a maximum capacity of 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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

> **Note**
> When an Amazon MSK cluster is the scalable target, scale in is disabled and cannot be enabled.

Amazon Neptune and Application Auto Scaling

You can scale Neptune clusters using target tracking scaling policies and scheduled scaling.

Use the following information to help you integrate Neptune with Application Auto Scaling.

If you are just getting started with scaling Neptune clusters, you can view sample configurations and details about using Neptune with Application Auto Scaling in the following documentation:

- Auto-scaling the number of replicas in an Amazon Neptune DB cluster in the *Neptune User Guide*

Service-linked role created for Neptune

The following service-linked role is automatically created in your AWS account when registering Neptune resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Neptune clusters as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for a Neptune cluster. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

To configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a Neptune cluster. The following example registers the desired capacity of a cluster called `mycluster`, with a minimum capacity of one and a maximum capacity of eight.

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

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Supported regions

Application Auto Scaling currently supports integration with Neptune in all commercial AWS Regions where Neptune is available, excluding AWS GovCloud (US) Regions.

Amazon SageMaker and Application Auto Scaling

You can scale SageMaker endpoint variants using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate SageMaker with Application Auto Scaling.

If you are just getting started with scaling SageMaker endpoint variants, you can view sample configurations and details about using SageMaker with Application Auto Scaling in the following documentation:

- Automatically scale Amazon SageMaker models in the *Amazon SageMaker Developer Guide*

Service-linked role created for SageMaker

The following service-linked role is automatically created in your AWS account when registering SageMaker resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering SageMaker endpoint variants as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for an SageMaker xxx. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the SageMaker console, then SageMaker automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for an SageMaker endpoint variant. The following example registers the desired EC2 instance count for a product variant called `my-variant`, running on the `my-endpoint` endpoint, with a minimum capacity of one instance and a maximum capacity of eight instances.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Amazon EC2 Spot Fleet and Application Auto Scaling

You can scale Spot Fleets using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate Spot Fleet with Application Auto Scaling.

If you are just getting started with scaling Spot Fleets, you can view details about using Spot Fleet with Application Auto Scaling in the following documentation:

- Automatic scaling for Spot Fleet in the *Amazon EC2 User Guide*

Service-linked role created for Spot Fleet

The following service-linked role is automatically created in your AWS account when registering Spot Fleet resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering Spot Fleets as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for a Spot Fleet. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

If you configure auto scaling using the Spot Fleet console, then Spot Fleet automatically registers a scalable target for you.

If you want to configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a Spot Fleet. The following example registers the target capacity of a Spot Fleet using its request ID, with a minimum capacity of two instances and a maximum capacity of 10 instances.

```
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
```

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Custom resources and Application Auto Scaling

You can scale custom resources using target tracking scaling policies, step scaling policies, and scheduled scaling.

Use the following information to help you integrate custom resources with Application Auto Scaling.

If you are just getting started with scaling custom resources, you can view our GitHub repository, which provides details about how custom resources integrate with Application Auto Scaling.

Service-linked role created for custom resources

The following service-linked role is automatically created in your AWS account when registering custom resources as scalable targets with Application Auto Scaling. This role allows Application Auto Scaling to perform supported operations within your account. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

- `AWSServiceRoleForApplicationAutoScaling_CustomResource`

Service principal used by the service-linked role

The service-linked role in the previous section can be assumed only by the service principal authorized by the trust relationships defined for the role. The service-linked role used by Application Auto Scaling grants access to the following service principal:

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

Registering custom resources as scalable targets with Application Auto Scaling

Application Auto Scaling requires a scalable target before you can create scaling policies or scheduled actions for a custom resource. A scalable target is a resource that Application Auto Scaling can scale out and scale in. Scalable targets are uniquely identified by the combination of resource ID, scalable dimension, and namespace.

To configure auto scaling using the AWS CLI or one of the AWS SDKs, you can use the following options:

- AWS CLI:

  Call the register-scalable-target command for a custom resource. The following example registers a custom resource as a scalable target, with a minimum desired count of one capacity unit and a maximum desired count of 10 capacity units. The `custom-resource-id.txt` file contains a string that identifies the resource ID, which represents the path to the custom resource through your Amazon API Gateway endpoint.

  ```
  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
  ```

  Contents of `custom-resource-id.txt`:

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

- AWS SDK:

  Call the RegisterScalableTarget operation and provide `ResourceId`, `ScalableDimension`, `ServiceNamespace`, `MinCapacity`, and `MaxCapacity` as parameters.

Target tracking scaling policies for Application Auto Scaling

With target tracking scaling policies, you choose a scaling metric and set a target value. Application Auto Scaling creates and manages the CloudWatch alarms that trigger the scaling policy and calculates the scaling adjustment based on the metric and the target value. The scaling policy adds or removes capacity as required to keep the metric at, or close to, the specified target value. In addition to keeping the metric close to the target value, a target tracking scaling policy also adjusts to changes in the metric due to a changing load pattern.

Choosing metrics

The following predefined metrics are available for use when you create target tracking scaling policies. You can optionally define which metric to monitor and use with your target tracking scaling policy by using a custom metric specification.

AppStream 2.0

- `AppStreamAverageCapacityUtilization`

Aurora

- `RDSReaderAverageCPUUtilization`
- `RDSReaderAverageDatabaseConnections`

Amazon Comprehend

- `ComprehendInferenceUtilization`

DynamoDB

- `DynamoDBReadCapacityUtilization`
- `DynamoDBWriteCapacityUtilization`

Amazon ECS

- `ALBRequestCountPerTarget` (load balancer metric)
- `ECSServiceAverageCPUUtilization`
- `ECSServiceAverageMemoryUtilization`

ElastiCache

- `ElastiCachePrimaryEngineCPUUtilization`
- `ElastiCacheReplicaEngineCPUUtilization`

- `ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage`

Amazon Keyspaces (for Apache Cassandra)

- `CassandraReadCapacityUtilization`
- `CassandraWriteCapacityUtilization`

Lambda

- `LambdaProvisionedConcurrencyUtilization`

Amazon Managed Streaming for Apache Kafka (MSK)

- `KafkaBrokerStorageUtilization`

Neptune

- `NeptuneReaderAverageCPUUtilization`

Spot Fleet (Amazon EC2)

- `ALBRequestCountPerTarget` (load balancer metric)
- `EC2SpotFleetRequestAverageCPUUtilization`
- `EC2SpotFleetRequestAverageNetworkIn`
- `EC2SpotFleetRequestAverageNetworkOut`

SageMaker

- `SageMakerVariantInvocationsPerInstance`

Each metric represents a time-ordered set of data points stored in Amazon CloudWatch. While most of the metrics in AWS are reported every minute by default, Amazon EC2 metrics are reported every five minutes by default. For an additional charge, you can enable detailed monitoring to get metric data for instances at a one minute frequency. To ensure a faster response to utilization changes, we recommend that you enable detailed monitoring. For more information, see Enable or turn off detailed monitoring for your instances in the *Amazon EC2 User Guide for Linux Instances*.

Keep the following in mind when choosing a metric:

- Not all metrics work for target tracking. This can be important when you are specifying a custom metric. The metric must be a valid utilization metric and describe how busy a scalable target is. The metric value must increase or decrease proportionally to the capacity of the scalable target so that the metric data can be used to proportionally scale the scalable target.
- To use the `ALBRequestCountPerTarget` metric, you must specify the `ResourceLabel` parameter to identify the target group that is associated with the metric.
- When a metric emits real 0 values to CloudWatch (for example, `ALBRequestCountPerTarget`), Application Auto Scaling can scale in to 0 when there is no traffic to your application. To have your scalable target scale in to 0 when no requests are routed it, the scalable target's minimum capacity must be set to 0.
- Not all services allow you to manage custom metrics through the console of the target service. To see if the target service supports custom metrics in the console, consult the documentation for that service.

Considerations

Keep the following considerations in mind:

- A target tracking scaling policy assumes that it should perform scale out when the specified metric is above the target value. You cannot use a target tracking scaling policy to scale out when the specified metric is below the target value.
- You may see gaps between the target value and the actual metric data points. This is because Application Auto Scaling always acts conservatively by rounding up or down when it determines how much capacity to add or remove. This prevents it from adding insufficient capacity or removing too much capacity. However, for a scalable target with a small capacity, the actual metric data points might seem far from the target value.
- For a scalable target with a larger capacity, adding or removing capacity causes less of a gap between the target value and the actual metric data points.
- To ensure application availability, Application Auto Scaling scales out proportionally to the metric as fast as it can, but scales in more gradually.
- You can have multiple target tracking scaling policies for a scalable target, provided that each of them uses a different metric. The intention of Application Auto Scaling is to always prioritize availability, so its behavior differs depending on whether the target tracking policies are ready for scale out or scale in. It will scale out the scalable target if any of the target tracking policies are ready for scale out, but will scale in only if all of the target tracking policies (with the scale-in portion enabled) are ready to scale in.
- If multiple policies instruct the scalable target to scale out or in at the same time, Application Auto Scaling scales based on the policy that provides the largest capacity for both scale in and scale out. This provides greater flexibility to cover multiple scenarios and ensures that there is always enough capacity to process your application workloads.
- You can disable the scale-in portion of a target tracking scaling policy. This feature provides the flexibility to use a different method for scale in than you use for scale out. For example, you can use a step scaling policy for scale in while using a target tracking scaling policy for scale out.
- We recommend caution, however, when using target tracking scaling policies with step scaling policies because conflicts between these policies can cause undesirable behavior. For example, if the step scaling policy initiates a scale-in activity before the target tracking policy is ready to scale in, the scale-in activity will not be blocked. After the scale-in activity completes, the target tracking policy could instruct the scalable target to scale out again.
- A target tracking scaling policy does not scale in when the state of the alarm is `INSUFFICIENT_DATA`. For more information, see Monitoring with CloudWatch alarms (p. 72).
- Do not edit or delete the CloudWatch alarms that are configured for the target tracking scaling policy. CloudWatch alarms that are associated with your target tracking scaling policies are managed by AWS and deleted automatically when no longer needed.

Cooldown period

The amount of time to wait for a previous scaling activity to take effect is called the cooldown period.

With target tracking scaling policies, there are two types of cooldown periods:

- With the *scale-out cooldown period*, the intention is to continuously (but not excessively) scale out. After Application Auto Scaling successfully scales out using a target tracking scaling policy, it starts to calculate the cooldown time. The scaling policy won't increase the desired capacity again unless either a larger scale out is triggered or the cooldown period ends. While the scale-out cooldown period is in effect, the capacity added by the initiating scale-out activity is calculated as part of the desired capacity for the next scale-out activity.

- With the *scale-in cooldown period*, the intention is to scale in conservatively to protect your application's availability, so scale-in activities are blocked until the cooldown period has expired. However, if another alarm triggers a scale-out activity during the scale-in cooldown period, Application Auto Scaling scales out the target immediately. In this case, the scale-in cooldown period stops and doesn't complete.

Each cooldown period is measured in seconds and applies only to scaling policy-related scaling activities. During a cooldown period, when a scheduled action starts at the scheduled time, it can trigger a scaling activity immediately without waiting for the cooldown period to expire.

You can start with the default values, which can be later fine-tuned. For example, you might need to increase a cooldown period to prevent your target tracking scaling policy from being too aggressive about changes that occur over short periods of time. For the default values, see TargetTrackingScalingPolicyConfiguration in the *Application Auto Scaling API Reference*.

Supporting application availability during high utilization periods

A target tracking scaling policy is more aggressive in adding capacity when utilization increases than it is in removing capacity when utilization decreases. For example, if the policy's specified metric reaches its target value, the policy assumes that your application is already heavily loaded. So it responds by adding capacity proportional to the metric value as fast as it can. The higher the metric, the more capacity is added.

When the metric falls below the target value, the policy expects that utilization will eventually increase again. So it slows down scaling by removing capacity only when utilization passes a threshold that is far enough below the target value (usually more than 10% lower) for utilization to be considered to have slowed. The intention of this more conservative behavior is to ensure that removing capacity only happens when the application is no longer experiencing demand at the same high level that it was previously. This is currently the default behavior for all target tracking scaling policies (though the behavior could change in the future).

For workloads that are cyclical in nature, you also have the option to automate capacity changes on a schedule using scheduled scaling. For each scheduled action, a new minimum capacity value and a new maximum capacity value can be defined. These values form the boundaries of the scaling policy.

The combination of scheduled scaling and target tracking scaling can help reduce the impact of a sharp increase in utilization levels, when capacity is needed immediately.

Commonly used commands for scaling policy creation, management, and deletion

The commonly used commands for working with scaling policies include:

- register-scalable-target to register AWS or custom resources as scalable targets (a resource that Application Auto Scaling can scale), and to suspend and resume scaling.
- put-scaling-policy to add or modify scaling policies for an existing scalable target.
- describe-scaling-activities to return information about scaling activities in an AWS Region.
- describe-scaling-policies to return information about scaling policies in an AWS Region.
- delete-scaling-policy to delete a scaling-policy.

Limitations

The following are limitations when using target tracking scaling policies:

- The scalable target can't be an Amazon EMR cluster. Target tracking scaling policies are not supported for Amazon EMR.
- When an Amazon MSK cluster is the scalable target, scale in is disabled and cannot be enabled.
- You cannot use the `RegisterScalableTarget` or `PutScalingPolicy` API operations to update an AWS Auto Scaling scaling plan. For information about using scaling plans, see the AWS Auto Scaling documentation.

Creating a target tracking scaling policy using the AWS CLI

You can create a target tracking scaling policy for Application Auto Scaling by using the AWS CLI for the following configuration tasks.

Tasks

Register a scalable target

If you haven't already done so, register the scalable target. Use the register-scalable-target command to register a specific resource in the target service as a scalable target. The following example registers a Spot Fleet request with Application Auto Scaling. Application Auto Scaling can scale the number of instances in the Spot Fleet at a minimum of 2 instances and a maximum of 10.

Linux, macOS, or Unix

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

Windows

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

Note
For brevity, the examples in this topic illustrate CLI commands for an Amazon EC2 Spot Fleet. To specify a different scalable target, specify its namespace in `--service-namespace`, its scalable dimension in `--scalable-dimension`, and its resource ID in `--resource-id`. For more information, see register-scalable-target.

Create a target tracking scaling policy

Example: target tracking configuration file

The following is an example target tracking configuration that keeps the average CPU utilization at 40 percent. Save this configuration in a file named `config.json`.

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

For more information, see PredefinedMetricSpecification in the *Application Auto Scaling API Reference*.

Alternatively, you can use a custom metric for scaling by creating a customized metric specification and adding values for each parameter from CloudWatch. The following is an example target tracking configuration that keeps the average utilization of the specified metric at 40 percent.

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

For more information, see CustomizedMetricSpecification in the *Application Auto Scaling API Reference*.

Example: cpu40-target-tracking-scaling-policy

Use the following put-scaling-policy command, along with the `config.json` file you created, to create a scaling policy named `cpu40-target-tracking-scaling-policy`.

Linux, macOS, or Unix

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

Windows

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

If successful, this command returns the ARNs and names of the two CloudWatch alarms created on your behalf.

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

Describe target tracking scaling policies

You can describe all scaling policies for the specified service namespace using the following describe-scaling-policies command.

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

You can filter the results to just the target tracking scaling policies using the `--query` parameter. For more information about the syntax for `query`, see Controlling command output from the AWS CLI in the *AWS Command Line Interface User Guide*.

Linux, macOS, or 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`]"
```

The following is example output.

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

Delete a target tracking scaling policy

When you are finished with a target tracking scaling policy, you can delete it using the delete-scaling-policy command.

The following command deletes the specified target tracking scaling policy for the specified Spot Fleet request. It also deletes the CloudWatch alarms that Application Auto Scaling created on your behalf.

Linux, macOS, or Unix

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

Windows

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

Step scaling policies for Application Auto Scaling

With step scaling, you choose scaling metrics and threshold values for the CloudWatch alarms that trigger the scaling process as well as define how your scalable target should be scaled when a threshold is in breach for a specified number of evaluation periods.

If your scaling metric is a utilization metric that increases or decreases proportionally to the capacity of the scalable target, we recommend that you use a target tracking scaling policy. For more information, see Target tracking scaling policies for Application Auto Scaling (p. 33). You still have the option to use target tracking scaling with step scaling for a more advanced scaling policy configuration. For example, if you want, you can configure a more aggressive response when utilization reaches a certain level.

Step scaling policies increase or decrease the current capacity of a scalable target based on a set of scaling adjustments, known as *step adjustments*. The adjustments vary based on the size of the alarm breach. All alarms that are breached are evaluated by Application Auto Scaling as it receives the alarm messages.

Step adjustments

When you create a step scaling policy, you add one or more step adjustments that enable you to scale based on the size of the alarm breach. Each step adjustment specifies the following:

- A lower bound for the metric value
- An upper bound for the metric value
- The amount by which to scale, based on the scaling adjustment type

CloudWatch aggregates metric data points based on the statistic for the metric associated with your CloudWatch alarm. When the alarm is breached, the appropriate scaling policy is triggered. Application Auto Scaling applies your specified aggregation type to the most recent metric data points from CloudWatch (as opposed to the raw metric data). It compares this aggregated metric value against the upper and lower bounds defined by the step adjustments to determine which step adjustment to perform.

You specify the upper and lower bounds relative to the breach threshold. For example, let's say that you have a scalable target that has both a current capacity and a desired capacity of 10. You have a CloudWatch alarm with a breach threshold of 50 percent. You have an adjustment type of `PercentChangeInCapacity` and scale-out and scale-in policies with the following step adjustments:

Example: Step adjustments for scale-out policy

Lower bound	Upper bound	Adjustment
0	10	0
10	20	10
20	null	30

Example: Step adjustments for scale-in policy

Lower bound	Upper bound	Adjustment
-10	0	0
-20	-10	-10
null	-20	-30

This creates the following scaling configuration.

```
Metric value

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

The following points summarize the behavior of the scaling configuration in relation to the desired and current capacity of the scalable target:

- The current and desired capacity is maintained while the aggregated metric value is greater than 40 and less than 60.
- If the metric value gets to 60, Application Auto Scaling increases the desired capacity of the scalable target by 1, to 11. That's based on the second step adjustment of the scale-out policy (add 10 percent of 10). After the new capacity is added, Application Auto Scaling increases the current capacity to 11. If the metric value rises to 70 even after this increase in capacity, Application Auto Scaling increases the target capacity by 3, to 14. That's based on the third step adjustment of the scale-out policy (add 30 percent of 11, 3.3, rounded down to 3).
- If the metric value gets to 40, Application Auto Scaling decreases the target capacity by 1, to 13, based on the second step adjustment of the scale-in policy (remove 10 percent of 14, 1.4, rounded down to 1). If the metric value falls to 30 even after this decrease in capacity, Application Auto Scaling decreases the target capacity by 3, to 10, based on the third step adjustment of the scale-in policy (remove 30 percent of 13, 3.9, rounded down to 3).

When you specify the step adjustments for your scaling policy, note the following:

- The ranges of your step adjustments can't overlap or have a gap.
- Only one step adjustment can have a null lower bound (negative infinity). If one step adjustment has a negative lower bound, then there must be a step adjustment with a null lower bound.
- Only one step adjustment can have a null upper bound (positive infinity). If one step adjustment has a positive upper bound, then there must be a step adjustment with a null upper bound.
- The upper and lower bound can't be null in the same step adjustment.
- If the metric value is above the breach threshold, the lower bound is inclusive and the upper bound is exclusive. If the metric value is below the breach threshold, the lower bound is exclusive and the upper bound is inclusive.

Scaling adjustment types

You can define a scaling policy that performs the optimal scaling action, based on the scaling adjustment type that you choose. You can specify the adjustment type as a percentage of the current capacity of your scalable target or in absolute numbers.

Application Auto Scaling supports the following adjustment types for step scaling policies:

- **ChangeInCapacity**—Increase or decrease the current capacity of the scalable target by the specified value. A positive value increases the capacity and a negative value decreases the capacity. For example: If the current capacity is 3 and the adjustment is 5, then Application Auto Scaling adds 5 to the capacity for a total of 8.
- **ExactCapacity**—Change the current capacity of the scalable target to the specified value. Specify a positive value with this adjustment type. For example: If the current capacity is 3 and the adjustment is 5, then Application Auto Scaling changes the capacity to 5.
- **PercentChangeInCapacity**—Increase or decrease the current capacity of the scalable target by the specified percentage. A positive value increases the capacity and a negative value decreases the capacity. For example: If the current capacity is 10 and the adjustment is 10 percent, then Application Auto Scaling adds 1 to the capacity for a total of 11.

  Note
  If the resulting value is not an integer, Application Auto Scaling rounds it as follows:

  - Values greater than 1 are rounded down. For example, `12.7` is rounded to `12`.
  - Values between 0 and 1 are rounded to 1. For example, `.67` is rounded to `1`.
  - Values between 0 and -1 are rounded to -1. For example, `-.58` is rounded to `-1`.
  - Values less than -1 are rounded up. For example, `-6.67` is rounded to `-6`.

  With **PercentChangeInCapacity**, you can also specify the minimum amount to scale using the `MinAdjustmentMagnitude` parameter. For example, suppose that you create a policy that adds 25 percent and you specify a minimum amount of 2. If the scalable target has a capacity of 4 and the scaling policy is performed, 25 percent of 4 is 1. However, because you specified a minimum increment of 2, Application Auto Scaling adds 2.

Cooldown period

The amount of time to wait for a previous scaling activity to take effect is called the cooldown period.

- With scale-out policies, the intention is to continuously (but not excessively) scale out. After Application Auto Scaling successfully scales out using a step scaling policy, it starts to calculate the cooldown time. The scaling policy won't increase the desired capacity again unless either a larger scale out is triggered or the cooldown period ends. While the cooldown period is in effect, capacity added by the initiating scale-out activity is calculated as part of the desired capacity for the next scale-out activity. For example, when an alarm triggers a step scaling policy to increase the capacity by 2, the scaling activity completes successfully, and a cooldown period starts. If the alarm triggers again during the cooldown period but at a more aggressive step adjustment of 3, the previous increase of 2 is considered part of the current capacity. Therefore, only 1 is added to the capacity.
- With scale-in policies, the intention is to scale in conservatively to protect your application's availability, so scale-in activities are blocked until the cooldown period has expired. However, if another alarm triggers a scale-out activity during the cooldown period after a scale-in activity, Application Auto Scaling scales out the target immediately. In this case, the cooldown period for the scale-in activity stops and doesn't complete.

The cooldown period is measured in seconds and applies only to scaling policy-related scaling activities. During a cooldown period, when a scheduled action starts at the scheduled time, it can trigger a scaling activity immediately without waiting for the cooldown period to expire.

Commonly used commands for scaling policy creation, management, and deletion

The commonly used commands for working with scaling policies include:

- register-scalable-target to register AWS or custom resources as scalable targets (a resource that Application Auto Scaling can scale), and to suspend and resume scaling.
- put-scaling-policy to add or modify scaling policies for an existing scalable target.
- describe-scaling-activities to return information about scaling activities in an AWS Region.
- describe-scaling-policies to return information about scaling policies in an AWS Region.
- delete-scaling-policy to delete a scaling-policy.

Limitations

The following are limitations when using step scaling policies:

- You cannot create step scaling policies for certain services. Step scaling policies are not supported for DynamoDB, Amazon Comprehend, Lambda, Amazon Keyspaces, Amazon MSK, ElastiCache, or Neptune.

Creating a step scaling policy using the AWS CLI

You can create a step scaling policy for Application Auto Scaling by using the AWS CLI for the following configuration tasks.

Tasks

Register a scalable target

If you haven't already done so, register the scalable target. Use the register-scalable-target command to register a specific resource in the target service as a scalable target. The following example registers an Amazon ECS service with Application Auto Scaling. Application Auto Scaling can scale the number of tasks at a minimum of 2 tasks and a maximum of 10.

Linux, macOS, or Unix

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

Windows

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

Note

For brevity, the examples in this topic illustrate CLI commands for an Amazon ECS service. To specify a different scalable target, specify its namespace in `--service-namespace`, its scalable dimension in `--scalable-dimension`, and its resource ID in `--resource-id`. For more information, see register-scalable-target.

Create a step scaling policy

The following is an example step configuration with an adjustment type of `ChangeInCapacity` that increases the capacity of the scalable target based on the following step adjustments (assuming a CloudWatch alarm threshold of 70 percent):

- Increase capacity by 1 when the value of the metric is greater than or equal to 70 percent but less than 85 percent
- Increase capacity by 2 when the value of the metric is greater than or equal to 85 percent but less than 95 percent
- Increase capacity by 3 when the value of the metric is greater than or equal to 95 percent

Save this configuration in a file named `config.json`.

```
{
  "AdjustmentType": "ChangeInCapacity",
  "MetricAggregationType": "Average",
  "Cooldown": 60,
  "StepAdjustments": [
    {
      "MetricIntervalLowerBound": 0,
      "MetricIntervalUpperBound": 15,
      "ScalingAdjustment": 1
    },
    {
      "MetricIntervalLowerBound": 15,
      "MetricIntervalUpperBound": 25,
      "ScalingAdjustment": 2
    },
    {
      "MetricIntervalLowerBound": 25,
      "ScalingAdjustment": 3
    }
  ]
}
```

Use the following put-scaling-policy command, along with the `config.json` file that you created, to create a scaling policy named `my-step-scaling-policy`.

Linux, macOS, or Unix

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

Windows

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

The output includes the ARN that serves as a unique name for the policy. You need it to create CloudWatch alarms.

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

Create an alarm that triggers the scaling policy

Finally, use the following CloudWatch put-metric-alarm command to create an alarm to use with your step scaling policy. In this example, you have an alarm based on average CPU utilization. The alarm is configured to be in an ALARM state if it reaches a threshold of 70 percent for at least two consecutive evaluation periods of 60 seconds. To specify a different CloudWatch metric or use your own custom metric, specify its name in `--metric-name` and its namespace in `--namespace`.

Linux, macOS, or Unix

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

Windows

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

Describe step scaling policies

You can describe all scaling policies for the specified service namespace using the following describe-scaling-policies command.

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

You can filter the results to just the step scaling policies using the `--query` parameter. For more information about the syntax for `query`, see Controlling command output from the AWS CLI in the *AWS Command Line Interface User Guide*.

Linux, macOS, or 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`]"
```

The following is example output.

```
[
    {
        "PolicyARN": "PolicyARN",
        "StepScalingPolicyConfiguration": {
            "MetricAggregationType": "Average",
            "Cooldown": 60,
            "StepAdjustments": [
                {
                    "MetricIntervalLowerBound": 0.0,
                    "MetricIntervalUpperBound": 15.0,
                    "ScalingAdjustment": 1
                },
                {
                    "MetricIntervalLowerBound": 15.0,
                    "MetricIntervalUpperBound": 25.0,
                    "ScalingAdjustment": 2
                },
                {
                    "MetricIntervalLowerBound": 25.0,
                    "ScalingAdjustment": 3
                }
            ],
            "AdjustmentType": "ChangeInCapacity"
        },
        "PolicyType": "StepScaling",
        "ResourceId": "service/default/sample-app-service",
        "ServiceNamespace": "ecs",
        "Alarms": [
            {
                "AlarmName": "Step-Scaling-AlarmHigh-ECS:service/default/sample-app-
service",
                "AlarmARN": "arn:aws:cloudwatch:region:012345678910:alarm:Step-Scaling-
AlarmHigh-ECS:service/default/sample-app-service"
            }
        ],
        "PolicyName": "my-step-scaling-policy",
        "ScalableDimension": "ecs:service:DesiredCount",
        "CreationTime": 1515024099.901
    }
]
```

Delete a step scaling policy

When you no longer need a step scaling policy, you can delete it. To delete both the scaling policy and the CloudWatch alarm, complete the following tasks.

To delete your scaling policy

Use the following delete-scaling-policy command.

Linux, macOS, or Unix

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

Windows

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

To delete the CloudWatch alarm

Use the delete-alarms command. You can delete one or more alarms at a time. For example, use the following command to delete the `Step-Scaling-AlarmHigh-ECS:service/default/sample-app-service` and `Step-Scaling-AlarmLow-ECS:service/default/sample-app-service` alarms.

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

Scheduled scaling for Application Auto Scaling

Scaling based on a schedule allows you to set your own scaling schedule according to predictable load changes. For example, let's say that every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can configure a schedule for Application Auto Scaling to increase capacity on Wednesday and decrease capacity on Friday.

To use scheduled scaling, create *scheduled actions*, which tell Application Auto Scaling to perform scaling activities at specific times. When you create a scheduled action, you specify the scalable target, when the scaling activity should occur, a minimum capacity, and a maximum capacity. You can create scheduled actions that scale one time only or that scale on a recurring schedule.

At the specified time, Application Auto Scaling scales based on the new capacity values, by comparing current capacity to the specified minimum and maximum capacity.

- If current capacity is less than the specified minimum capacity, Application Auto Scaling scales out (increases capacity) to the specified minimum capacity.
- If current capacity is greater than the specified maximum capacity, Application Auto Scaling scales in (decreases capacity) to the specified maximum capacity.

You can use scheduled scaling and scaling policies together on the same resource to get the benefits of both. After a scheduled action runs, the scaling policy can continue to make decisions about whether to further scale capacity. This helps you ensure that you have sufficient capacity to handle the load for your application. While your application scales to match demand, current capacity must fall within the minimum and maximum capacity that was set by your scheduled action.

For a detailed example of using scheduled scaling, see the blog post Scheduling AWS Lambda Provisioned Concurrency for recurring peak usage on the AWS Compute Blog. For a tutorial that walks through how to create scheduled actions using sample AWS resources, see Getting started using the AWS CLI (p. 5).

Considerations

When you create a scheduled action, keep the following in mind:

- A scheduled action sets the `MinCapacity` and `MaxCapacity` to what is specified by the scheduled action at the time specified.
- By default, the times that you set are in Coordinated Universal Time (UTC). When specifying a one-time schedule or a recurring schedule with a cron expression, you can change the time zone to correspond to your local time zone or a time zone for another part of your network. When you specify a time zone that observes daylight saving time, it automatically adjusts for Daylight Saving Time (DST).
- If you specify a recurring schedule, you can specify a date and time for the start time, the end time, or both.
  - The start time and end time must be set in UTC. Changing the time zone for start and end time is not supported.
  - If you specify a start time, Application Auto Scaling performs the action at that time, and then performs the action based on the specified recurrence.

  - If you specify an end time, the action stops repeating after this time. Application Auto Scaling does not keep track of previous values and revert back to those previous values after the end time.
- The name of the scheduled action must be unique among all other scheduled actions on the specified scalable target.
- Application Auto Scaling guarantees the order in which scheduled actions run for the same scalable target, but cannot guarantee the order in which scheduled actions run across scalable targets.
- Due to the distributed nature of Application Auto Scaling and the target services, the delay between the time the scheduled action is triggered and the time the target service honors the scaling action might be a few seconds. Because scheduled actions are run in the order that they are specified, scheduled actions with start times close to each other can take longer to run.

Commonly used commands for scheduled action creation, management, and deletion

The commonly used commands for working with schedule scaling include:

- register-scalable-target to register AWS or custom resources as scalable targets (a resource that Application Auto Scaling can scale), and to suspend and resume scaling.
- put-scheduled-action to add or modify scheduled actions for an existing scalable target.
- describe-scaling-activities to return information about scaling activities in an AWS Region.
- describe-scheduled-actions to return information about scheduled actions in an AWS Region.
- delete-scheduled-action to delete a scheduled action.

Note
For brevity, the examples in this guide illustrate CLI commands for a few of the services that integrate with Application Auto Scaling. To specify a different scalable target, specify its namespace in `--service-namespace`, its scalable dimension in `--scalable-dimension`, and its resource ID in `--resource-id`. For more information, see register-scalable-target.

Limitations

The following are limitations when using scheduled scaling:

- Application Auto Scaling doesn't provide second-level precision in schedule expressions. The finest resolution using a cron expression is 1 minute.
- The scalable target can't be an Amazon MSK cluster. Scheduled scaling is not supported for Amazon MSK.

Example scheduled actions for Application Auto Scaling

The following examples show how to create scheduled actions with the AWS CLI put-scheduled-action command. When you specify the new capacity, you can specify a minimum capacity, a maximum capacity, or both.

Topics

Creating a scheduled action that occurs only once

To automatically scale your scalable target one time only, at a specified date and time, use the `--schedule "at(yyyy-mm-ddThh:mm:ss)"` option.

Example Example: To scale out one time only

The following is an example of creating a scheduled action to scale out capacity at a specific date and time.

At the date and time specified for `--schedule` (10:00 PM UTC on March 31, 2021), if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`.

Linux, macOS, or Unix

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

Windows

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

Note
When this scheduled action runs, if the maximum capacity is less than the value specified for minimum capacity, you must specify a new minimum and maximum capacity, and not just the minimum capacity.

Example Example: To scale in one time only

The following is an example of creating a scheduled action to scale in capacity at a specific date and time.

At the date and time specified for `--schedule` (10:30 PM UTC on March 31, 2021), if the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

Linux, macOS, or 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
```

Creating a scheduled action that runs on a recurring interval

To schedule scaling at a recurring interval, use the `--schedule "rate(value unit)"` option. The value must be a positive integer. The unit can be `minute`, `minutes`, `hour`, `hours`, `day`, or `days`. For more information, see Rate expressions in the *Amazon CloudWatch Events User Guide*.

The following is an example of a scheduled action that uses a rate expression.

On the specified schedule (every 5 hours starting on January 30, 2021 at 12:00 PM UTC and ending on January 31, 2021 at 10:00 PM UTC), if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`. If the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

Linux, macOS, or Unix

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

Windows

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

Creating a scheduled action that runs on a recurring schedule

To schedule scaling on a recurring schedule, use the `--schedule "cron(fields)"` option. The cron format that's supported by Application Auto Scaling consists of six fields separated by white spaces: [Minutes] [Hours] [Day_of_Month] [Month] [Day_of_Week] [Year].

Here are some examples of cron expressions.

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
0	10	*	*	?	*	Run at 10:00 am (UTC) every day

Minutes	Hours	Day of month	Month	Day of week	Year	Meaning
15	12	*	*	?	*	Run at 12:15 pm (UTC) every day
0	18	?	*	MON-FRI	*	Run at 6:00 pm (UTC) every Monday through Friday
0	8	1	*	?	*	Run at 8:00 am (UTC) the 1st day of every month
0/15	*	*	*	?	*	Run every 15 minutes
0/10	*	?	*	MON-FRI	*	Run every 10 minutes Monday through Friday
0/5	8-17	?	*	MON-FRI	*	Run every 5 minutes Monday through Friday between 8:00 am and 5:55 pm (UTC)

For more information about writing cron expressions, see Cron expressions in the *Amazon CloudWatch Events User Guide*.

The following is an example of a scheduled action that uses a cron expression.

On the specified schedule (every day at 9:00 AM UTC), if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`. If the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

Linux, macOS, or 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
```

Creating a one-time scheduled action that specifies a time zone

Scheduled actions are set to the UTC time zone by default. To specify a different time zone, include the `--timezone` option and specify the canonical name for the time zone (`America/New_York`, for example). For more information, see https://www.joda.org/joda-time/timezones.html, which provides information about the IANA time zones that are supported when calling put-scheduled-action.

The following is an example that uses the `--timezone` option when creating a scheduled action to scale capacity at a specific date and time.

At the date and time specified for `--schedule` (5:00 PM local time on January 31, 2021), if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`. If the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

Linux, macOS, or 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
```

Creating a recurring scheduled action that specifies a time zone

The following is an example that uses the `--timezone` option when creating a recurring scheduled action to scale capacity.

On the specified schedule (every Monday through Friday at 6:00 PM local time), if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`. If the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

Linux, macOS, or 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
```

Managing scheduled scaling for Application Auto Scaling

The AWS CLI includes several other commands that help you manage your scheduled actions.

Topics

Viewing scaling activities for a specified service

To view the scaling activities for all of the scalable targets in a specified service namespace, use the describe-scaling-activities command.

The following example retrieves the scaling activities associated with the `dynamodb` service namespace.

Linux, macOS, or Unix

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

Windows

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

If the command succeeds, you see output similar to the following.

```
{
    "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"
        }
    ]
}
```

To change this command so that it retrieves the scaling activities for only one of your scalable targets, add the `--resource-id` option.

Describing all scheduled actions for a specified service

To describe the scheduled actions for all of the scalable targets in a specified service namespace, use the describe-scheduled-actions command.

The following example retrieves the scheduled actions associated with the `ec2` service namespace.

Linux, macOS, or Unix

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

Windows

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

If successful, this command returns output similar to the following.

```
{
    "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
        }
    ]
}
```

Describing one or more scheduled actions for a scalable target

To retrieve information about the scheduled actions for a specified scalable target, add the `--resource-id` option when describing scheduled actions using the describe-scheduled-actions command.

If you include the `--scheduled-action-names` option and specify the name of a scheduled action as its value, the command returns only the scheduled action whose name is a match, as shown in the following example.

Linux, macOS, or 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
```

The following is example output.

```
{
    "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
        }
    ]
}
```

If more than one value is provided for the `--scheduled-action-names` option, all scheduled actions whose names are a match are included in the output.

Turning off scheduled scaling for a scalable target

You can temporarily turn off scheduled scaling without deleting your scheduled actions. For more information, see Suspending and resuming scaling for Application Auto Scaling (p. 67).

Suspend scheduled scaling on a scalable target by using the register-scalable-target command with the `--suspended-state` option, and specifying `true` as the value of the `ScheduledScalingSuspended` attribute, as shown in the following example.

Linux, macOS, or 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}"
```

If successful, this command returns to the prompt.

To resume scheduled scaling, run this command again, specifying `false` as the value of the `ScheduledScalingSuspended` attribute.

Deleting a scheduled action

When you are finished with a scheduled action, you can delete it using the delete-scheduled-action command.

Linux, macOS, or 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
```

If successful, this command returns to the prompt.

Tutorial: Configuring auto scaling to handle a heavy workload

In this tutorial, you learn how to scale out and in based on time windows when your application will have a heavier than normal workload. This is helpful when you have an application that can suddenly have a large number of visitors on a regular schedule or on a seasonal basis.

You can use a target tracking scaling policy together with scheduled scaling to handle the extra load. Scheduled scaling automatically initiates changes to your `MinCapacity` and `MaxCapacity` on your behalf, based on a schedule that you specify. When a target tracking scaling policy is active on the resource, it can scale dynamically based on current resource utilization, within the new minimum and maximum capacity range.

After completing this tutorial, you'll know how to:

- Use scheduled scaling to add extra capacity to meet a heavy load before it arrives, and then remove the extra capacity when it's no longer required.
- Use a target tracking scaling policy to scale your application based on current resource utilization.

Contents

Prerequisites

This tutorial assumes that you have already done the following:

- You created an AWS account.
- You installed and configured the AWS CLI. For more information, see Set up the AWS CLI (p. 4).
- Your account has all of the necessary permissions for registering and deregistering resources as a scalable target with Application Auto Scaling. It also has all of the necessary permissions for creating scaling policies and scheduled actions.
- You have a scalable resource in a non-production environment available to use for this tutorial. If you don't already have one, create one before beginning the tutorial.

Before you begin, note the following:

While completing this tutorial, there are two steps in which you set or update your `MinCapacity` and `MaxCapacity` values to 0 to reset the current capacity to 0. Depending on the resource that you've chosen to use, you might be unable to reset the current capacity to 0 during these steps. To help you address the issue, a message in the output will indicate that minimum capacity cannot be less than the value specified and will provide the minimum capacity value that the resource can accept.

To monitor your scaling activities with Application Auto Scaling, you can use the describe-scaling-activities command. Each scaling event that is triggered by a scaling policy or a scheduled action generates a scaling activity.

Step 1: Register your scalable target

Start by registering your resource as a scalable target with Application Auto Scaling. A scalable target is a resource that Application Auto Scaling can scale out or scale in.

To register your scalable target with Application Auto Scaling

- Use the following register-scalable-target command to register a new scalable target. Set the `MinCapacity` and `MaxCapacity` values to 0 to reset the current capacity to 0.

  Linux, macOS, or Unix

  ```
  aws application-autoscaling register-scalable-target \
    --service-namespace service namespace \
    --scalable-dimension scalable dimension of the scalable target \
    --resource-id resource identifier to associate with this scalable target \
    --min-capacity 0 --max-capacity 0
  ```

  Windows

  ```
  aws application-autoscaling register-scalable-target --service-namespace service
   namespace --scalable-dimension scalable dimension of the scalable target --resource-
  id resource identifier to associate with this scalable target --min-capacity 0 --max-
  capacity 0
  ```

  This command does not return any output if it is successful.

Step 2: Set up scheduled actions according to your requirements

You can use the put-scheduled-action command to create scheduled actions that are configured to meet your business needs. In this tutorial, we focus on a configuration that stops consuming resources outside of working hours by reducing capacity to 0.

To create a scheduled action that scales out in the morning

1. To scale out the scalable target, use the following put-scheduled-action command. Include the `--schedule` parameter with a recurring schedule, in UTC, using a cron expression.

   On the specified schedule (every day at 9:00 AM UTC), Application Auto Scaling updates the `MinCapacity` and `MaxCapacity` values to the desired range of 1-5 capacity units.

   Linux, macOS, or 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
```

This command does not return any output if it is successful.

2. To confirm that your scheduled action exists, use the following describe-scheduled-actions command.

Linux, macOS, or 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`]"
```

The following is example output.

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

To create a scheduled action that scales in at night

1. Repeat the preceding procedure to create another scheduled action that Application Auto Scaling uses to scale in at the end of the day.

   On the specified schedule (every day at 8:00 PM UTC), Application Auto Scaling updates the target's `MinCapacity` and `MaxCapacity` to 0, as instructed by the following put-scheduled-action command.

Linux, macOS, or 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. To confirm that your scheduled action exists, use the following describe-scheduled-actions command.

   Linux, macOS, or 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`]"
```

   The following is example output.

```
[
    {
        "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
        },
        ...
    }
]
```

Step 3: Add a target tracking scaling policy

Now that you have the basic schedule in place, add a target tracking scaling policy to scale based on current resource utilization.

With target tracking, Application Auto Scaling compares the target value in the policy to the current value of the specified metric. When they are unequal for a period of time, Application Auto Scaling adds or removes capacity to maintain steady performance. As the load on your application and the metric value increases, Application Auto Scaling adds capacity as fast as it can without going above `MaxCapacity`. When Application Auto Scaling removes capacity because the load is minimal, it does so without going below `MinCapacity`. By adjusting the capacity based on usage, you only pay for what your application needs. For more information, see Supporting application availability during high utilization periods (p. 36).

If the metric has insufficient data because your application does not have any load, Application Auto Scaling does not add or remove capacity. The intention of this behavior is to prioritize availability in situations where not enough information is available.

You can add multiple scaling policies, but make sure you do not add conflicting step scaling policies, which might cause undesirable behavior. For example, if the step scaling policy initiates a scale-in activity before the target tracking policy is ready to scale in, the scale-in activity will not be blocked. After the scale-in activity completes, the target tracking policy could instruct Application Auto Scaling to scale out again.

To create a target tracking scaling policy

1. Use the following put-scaling-policy command to create the policy.

   The metrics that are most frequently used for target tracking are predefined, and you can use them without supplying the full metric specification from CloudWatch. For more information about the available predefined metrics, see Target tracking scaling policies for Application Auto Scaling (p. 33).

   Before you run this command, make sure that your predefined metric expects the target value. For example, to scale out when CPU reaches 50% utilization, specify a target value of 50.0. Or, to scale out Lambda provisioned concurrency when usage reaches 70% utilization, specify a target value of 0.7. For information about target values for a particular resource, refer to the documentation that is provided by the service about how to configure target tracking. For more information, see AWS services that you can use with Application Auto Scaling (p. 15).

   Linux, macOS, or 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\" }}"
   ```

   If successful, this command returns the ARNs and names of the two CloudWatch alarms that were created on your behalf.

2. To confirm that your scheduled action exists, use the following describe-scaling-policies command.

   Linux, macOS, or 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`]"
```

The following is example output.

```
[
    {
        "PolicyARN": "arn",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "predefinedmetric"
            },
            "TargetValue": 50.0
        },
        "PolicyName": "my-scaling-policy",
        "PolicyType": "TargetTrackingScaling",
        "Alarms": [],
        ...
    }
]
```

Step 4: Clean up

To prevent your account from accruing charges for resources created while actively scaling, you can clean up the associated scaling configuration as follows.

Deleting the scaling configuration does not delete your scalable resource. It also does not return it to its original capacity. You can use the console of the service where you created the scalable resource to delete it or adjust its capacity.

To delete the scheduled actions

The following delete-scheduled-action command deletes a specified scheduled action. You can skip this step if you want to keep the scheduled actions that you created.

Linux, macOS, or 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
```

To delete the scaling policy

The following delete-scaling-policy command deletes a specified target tracking scaling policy. You can skip this step if you want to keep the scaling policy that you created.

Linux, macOS, or 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
```

To deregister the scalable target

Use the following deregister-scalable-target command to deregister the scalable target. If you have any scaling policies that you created or any scheduled actions that have not yet been deleted, they are deleted by this command. You can skip this step if you want to keep the scalable target registered for future use.

Linux, macOS, or 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
```

Suspending and resuming scaling for Application Auto Scaling

This topic explains how to suspend and then resume one or more of the scaling activities for the scalable targets in your application. The suspend-resume feature is used to temporarily pause scaling activities triggered by your scaling policies and scheduled actions. This can be useful, for example, when you don't want automatic scaling to potentially interfere while you are making a change or investigating a configuration issue. Your scaling policies and scheduled actions can be retained, and when you are ready, scaling activities can be resumed.

In the example CLI commands that follow, you pass the JSON-formatted parameters in a config.json file. You can also pass these parameters on the command line by using quotation marks to enclose the JSON data structure. For more information, see Using quotation marks with strings in the AWS CLI in the *AWS Command Line Interface User Guide*.

Contents

Scaling activities

Application Auto Scaling supports putting the following scaling activities in a suspended state:

- All scale-in activities that are triggered by a scaling policy.
- All scale-out activities that are triggered by a scaling policy.
- All scaling activities that involve scheduled actions.

The following descriptions explain what happens when individual scaling activities are suspended. Each one can be suspended and resumed independently. Depending on the reason for suspending a scaling activity, you might need to suspend multiple scaling activities together.

`DynamicScalingInSuspended`

- Application Auto Scaling does not remove capacity when a target tracking scaling policy or a step scaling policy is triggered. This allows you to temporarily disable scale-in activities associated with scaling policies without deleting the scaling policies or their associated CloudWatch alarms. When you resume scale in, Application Auto Scaling evaluates policies with alarm thresholds that are currently in breach.

`DynamicScalingOutSuspended`

- Application Auto Scaling does not add capacity when a target tracking scaling policy or a step scaling policy is triggered. This allows you to temporarily disable scale-out activities associated with scaling policies without deleting the scaling policies or their associated CloudWatch alarms. When you resume scale out, Application Auto Scaling evaluates policies with alarm thresholds that are currently in breach.

`ScheduledScalingSuspended`

- Application Auto Scaling does not initiate the scaling actions that are scheduled to run during the suspension period. When you resume scheduled scaling, Application Auto Scaling only evaluates scheduled actions whose execution time has not yet passed.

Suspend and resume scaling activities using the AWS CLI

You can suspend and resume individual scaling activities or all scaling activities for your Application Auto Scaling scalable target.

Note
For brevity, these examples illustrate how to suspend and resume scaling for a DynamoDB table. To specify a different scalable target, specify its namespace in `--service-namespace`, its scalable dimension in `--scalable-dimension`, and its resource ID in `--resource-id`. For a list of valid values for each option, see register-scalable-target.

To suspend a scaling activity

Open a command-line window and use the register-scalable-target command with the `--suspended-state` option as follows.

Linux, macOS, or 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
```

To only suspend scale-in activities that are triggered by a scaling policy, specify the following in config.json.

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

To only suspend scale-out activities that are triggered by a scaling policy, specify the following in config.json.

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

To only suspend scaling activities that involve scheduled actions, specify the following in config.json.

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

To suspend all scaling activities

Use the register-scalable-target command with the `--suspended-state` option as follows.

Linux, macOS, or 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
```

This example assumes that the file config.json contains the following JSON-formatted parameters.

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

View suspended scaling activities

Use the describe-scalable-targets command to determine which scaling activities are in a suspended state for a scalable target.

Linux, macOS, or 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
```

The following is example output.

```
{
    "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"
        }
    ]
}
```

Resume scaling activities

When you are ready to resume the scaling activity, you can resume it using the register-scalable-target command.

The following example command resumes all scaling activities for the specified scalable target.

Linux, macOS, or 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
```

This example assumes that the file config.json contains the following JSON-formatted parameters.

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

Application Auto Scaling monitoring

Monitoring is an important part of maintaining the reliability, availability, and performance of Application Auto Scaling and your other AWS solutions. You should collect monitoring data from all parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides monitoring tools to watch Application Auto Scaling, report when something is wrong, and take automatic actions when appropriate.

You can use the following features to help you manage your AWS resources:

Amazon CloudWatch Alarms

To detect unhealthy application behavior, CloudWatch helps you by automatically monitoring certain metrics for your AWS resources. You can configure a CloudWatch alarm and set up an Amazon SNS notification that sends an email when a metric's value is not what you expect or when certain anomalies are detected. For example, you can be notified when network activity is suddenly higher or lower than a metric's expected value. For more information, see Monitoring with CloudWatch alarms (p. 72) and the Amazon CloudWatch User Guide.

Amazon CloudWatch Dashboards

Amazon CloudWatch monitors your AWS resources and the applications that you run on AWS in real time. CloudWatch dashboards are customizable home pages in the CloudWatch console. You can use these pages to monitor your resources in a single view, even including resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources. For more information, see Building dashboards with CloudWatch (p. 73).

Amazon EventBridge

EventBridge delivers a near real time stream of system events that describe changes in AWS resources. EventBridge enables automated event-driven computing. You can write rules that watch for certain events and trigger automated actions in other AWS when these events happen. For more information, see Getting notified of events preventing scaling through EventBridge (p. 75).

AWS CloudTrail

AWS CloudTrail captures API calls and related events made by or on behalf of your AWS account. Then it delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide. For information about the Application Auto Scaling API calls that are logged by CloudTrail, see Logging Application Auto Scaling API calls with CloudTrail.

Amazon CloudWatch Logs

Amazon CloudWatch Logs enable you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.

AWS Personal Health Dashboard

The AWS Personal Health Dashboard (PHD) displays information, and also provides notifications that are triggered by changes in the health of AWS resources. The information is presented in two ways: on a dashboard that shows recent and upcoming events organized by category, and in a full event log that shows all events from the past 90 days. For more information, see AWS Personal Health Dashboard notifications for Application Auto Scaling (p. 77).

Monitoring with CloudWatch alarms

You can create alarms to notify you when Amazon CloudWatch has detected any problems that might require your attention. CloudWatch helps you by automatically monitoring certain metrics for AWS.

A CloudWatch alarm watches a single metric. It invokes one or more actions only when the alarm state changes, and has persisted for the period that you specify. For example, you can set an alarm that notifies you when a metric value falls to or exceeds a certain level, ensuring that you are notified before a potential problem occurs.

CloudWatch also allows you to set an alarm that notifies you when the metric is in `INSUFFICIENT_DATA` state. Any metric, for any AWS service, can alarm on `INSUFFICIENT_DATA`. This is the initial state of a new alarm, but the alarm state also changes to `INSUFFICIENT_DATA` if CloudWatch metrics become unavailable, or not enough data is available for the metric to determine the alarm state. For example, AWS Lambda emits the `ProvisionedConcurrencyUtilization` metric to CloudWatch every minute only when the Lambda function is active. If the function is inactive, this results in the alarm going to the `INSUFFICIENT_DATA` state while waiting for the metrics. This is normal and might not necessarily mean that there is a problem, but it could be indicative of a problem if you expected activity within a period of time but there was none.

This topic explains how to create an alarm that sends a notification when the metric is within or outside a threshold that you define, or when there is insufficient data. For more detailed information about alarms, see Using Amazon CloudWatch alarms in the *Amazon CloudWatch User Guide*.

To create an alarm that sends email

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric**.

   You are directed to a page where you can find all of your metrics. The types of metrics available to you depends on the services and features that you use. Metrics are grouped first by the service namespace, then by the various dimension combinations within each namespace.
4. Select a metric namespace (for example, **Lambda**) and then a metric dimension (for example, **By Function Name**).

   The **All metrics** tab displays all metrics for the selected dimension and namespace.
5. Select the check box next to the metric that you want to create an alarm for, and then choose **Select metric**.
6. Configure the alarm as follows, and then choose **Next**:

   - Under **Metric**, select an aggregation period of `1 minute` or `5 minutes`. If you use one minute as an aggregation period for a metric, there will be one data point every minute. The shorter period creates a more sensitive alarm.
   - Under **Conditions**, configure your threshold, for example, the value that the metric must exceed before a notification is generated.
   - Under **Additional configuration**, for **Datapoints to alarm**, enter the number of data points (evaluation periods) during which the metric value must meet the threshold conditions to trigger the alarm. For example, two consecutive periods of 5 minutes would take 10 minutes to trigger the alarm.
   - For **Missing data treatment**, keep the default and treat missing data points as missing.

     Some metrics are reported only when there is activity occurring. This can result in a sparsely reported metric. If a metric is frequently missing data points by design, the state of the alarm is `INSUFFICIENT_DATA` during those periods. To force the alarm to maintain the previous `ALARM` or `OK` state to keep alerts from flapping, you could choose to ignore missing data instead.

7. Under **Notification**, choose or create an SNS topic to notify when the alarm is in `ALARM` state, `OK` state, or `INSUFFICIENT_DATA` state. To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.
8. When finished, choose **Next**.
9. Enter a name and, optionally, a description for the alarm, and then choose **Next**.
10. Choose **Create alarm**.

To check the state of your alarms

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms** to see a list of alarms.
3. To filter alarms, use the drop-down filters next to the search field, and choose the filter option that you want to apply.
4. To edit or delete an alarm, select the alarm and then choose **Actions**, **Edit** or **Actions**, **Delete**.

Building dashboards with CloudWatch

You can monitor how your application uses resources by using Amazon CloudWatch, which generates metrics about your usage and performance. CloudWatch collects raw data from your AWS resources and the applications that you run on AWS, and processes it into readable, near real time metrics. The metrics are kept for 15 months so that you can access historical information to gain a better perspective on how your application is performing. For more information, see the Amazon CloudWatch User Guide.

CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of selected metrics for your AWS resources. You can select the color used for each metric on each graph, so you can more easily track the same metric across multiple graphs.

To create a CloudWatch dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboard**, and then choose **Create new dashboard**.
3. Enter a name for the dashboard, such as the name of the service for which you want to view CloudWatch data.
4. Choose **Create dashboard**.
5. Choose a type of widget to add to your dashboard, such as a line graph. Then choose **Configure**, and choose the metric that you want to add to your dashboard. For more information, see Add or remove a graph from a CloudWatch dashboard in the *Amazon CloudWatch User Guide*

By default, the metrics that you create in the CloudWatch dashboards are averages.

Metrics and dimensions

When you interact with the services that integrate with Application Auto Scaling, they send the metrics shown in the following table to CloudWatch. In CloudWatch, metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

These metrics can help you discover your application's capacity requirements. You can use this information to set your capacity statically, or to set up automatic scaling. If your application's workload is not constant, this indicates that you should consider using automatic scaling.

Metric name	Namespace	Dimensions	Applies to	
AvailableCapacity	AWS/AppStream	Fleet	AppStream	
CapacityUtilization	AWS/AppStream	Fleet	AppStream	
CPUUtilization	AWS/RDS	DBClusterIdentifier, Role (READER)	Aurora	
DatabaseConnections	AWS/RDS	DBClusterIdentifier, Role (READER)	Aurora	
InferenceUtilization	AWS/Comprehend	EndpointArn	Comprehend	
ProvisionedReadCapacityUnits	AWS/DynamoDB	TableName, GlobalSecondaryIndexName	DynamoDB	
ProvisionedWriteCapacityUnits	AWS/DynamoDB	TableName, GlobalSecondaryIndexName	DynamoDB	
ConsumedReadCapacityUnits	AWS/DynamoDB	TableName, GlobalSecondaryIndexName	DynamoDB	
ConsumedWriteCapacityUnits	AWS/DynamoDB	TableName, GlobalSecondaryIndexName	DynamoDB	
CPUUtilization	AWS/ECS	ClusterName, ServiceName	ECS	
MemoryUtilization	AWS/ECS	ClusterName, ServiceName	ECS	
RequestCountPerTarget	AWS/ApplicationELB	TargetGroup	ECS	
YARNMemoryAvailablePercentage	AWS/ElasticMapReduce	ClusterId	EMR	
ProvisionedReadCapacityUnits	AWS/Cassandra	Keyspace, TableName	Amazon Keyspaces	
ProvisionedWriteCapacityUnits	AWS/Cassandra	Keyspace, TableName	Amazon Keyspaces	
ConsumedReadCapacityUnits	AWS/Cassandra	Keyspace, TableName	Amazon Keyspaces	
ConsumedWriteCapacityUnits	AWS/Cassandra	Keyspace, TableName	Amazon Keyspaces	
ProvisionedConcurrencyUtilization	AWS/Lambda	FunctionName, Resource	Lambda	
KafkaDataLogsDiskUsed	AWS/Kafka	Cluster Name	Amazon MSK	

Metric name	Namespac	Dimensions	Applies to	
KafkaDataLogsDiskUsed	AWS/ Kafka	Cluster Name, Broker ID	Amazon MSK	
InvocationsPerInstance	AWS/ SageMaker	EndpointName, VariantName	SageMaker	
CPUUtilization	AWS/ EC2Spot	FleetRequestId	Spot Fleet	
NetworkIn	AWS/ EC2Spot	FleetRequestId	Spot Fleet	
NetworkOut	AWS/ EC2Spot	FleetRequestId	Spot Fleet	
RequestCountPerTarget	AWS/ ApplicationELB	TargetGroup	Spot Fleet	

While CloudWatch allows you to choose any statistic and period for each metric, not all combinations are useful. For example, the Average, Minimum, and Maximum statistics for CPU utilization are useful, but the Sum statistic is not. For more information, refer to the service's documentation by following the links provided in the preceding table.

A commonly used measure of application performance is average CPU utilization. If there is an increase in CPU utilization and you have insufficient capacity to handle it, the application might become unresponsive. On the other hand, if you have too much capacity and resources are running when utilization is low, this increases the costs for using that service.

Depending on the service, you also have metrics that track the amount of provisioned throughput that is available. For example, for the number of invocations that are being processed on a function alias or version with provisioned concurrency, Lambda emits the `ProvisionedConcurrencyUtilization` metric. If you are starting a large job and invoke the same function many times simultaneously, the job might experience latency when it exceeds the amount of provisioned concurrency available. On the other hand, if you have more provisioned concurrency than you need, your costs might be higher than they should be.

If you do not see these metrics in the CloudWatch console, make sure that you have completed the set up of the resource. Metrics do not appear before the resource has been set up completely. Also, if a metric hasn't published data in the past 14 days, you can't find it when searching for metrics to add to a graph on a CloudWatch dashboard. For information about how to add any metric manually, see Graph metrics manually on a CloudWatch dashboard in the *Amazon CloudWatch User Guide*.

Getting notified of events preventing scaling through EventBridge

Application Auto Scaling integrates with Amazon EventBridge to notify you of certain events that affect scaling. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. The actions that can be automatically triggered include the following:

- Invoking an AWS Lambda function
- Invoking an Amazon EC2 Run Command

- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine
- Notifying an Amazon SNS topic or an Amazon SQS queue

You can also create a rule that triggers on an Application Auto Scaling API call. For more information, see Creating an EventBridge rule that triggers on an AWS API call using AWS CloudTrail in the *Amazon EventBridge User Guide*.

For more information, see Getting started with Amazon EventBridge in the *Amazon EventBridge User Guide*.

Application Auto Scaling events

The following are example events from Application Auto Scaling. Events are emitted on a best effort basis.

Only events that are specific to `scaledToMax` are currently available.

Event for State Change: Scaled to Max

The following event is sent when you reach the maximum capacity limit that you specified in the scaling configuration for that resource.

```
{
  "version": "0",
  "id": "7bf73129-1428-4cd3-a780-95db273d1602",
  "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": "2019-06-12T10:20:43Z",
    "endTime": "2019-06-12T10:23:40Z",
    "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",
}
```

Sample Event Pattern for Scaled to Max Event

Rules use event patterns to select events and route them to targets. The following is a sample Application Auto Scaling event pattern.

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

AWS Personal Health Dashboard notifications for Application Auto Scaling

To help you manage failed scaling events, your AWS Personal Health Dashboard provides support for notifications that are emitted by Application Auto Scaling. Only scale out events that are specific to your DynamoDB resources are currently available.

The AWS Personal Health Dashboard is part of the AWS Health service. It requires no setup and can be viewed by any user that is authenticated in your account. For more information, see Getting started with the AWS Personal Health Dashboard.

If your DynamoDB resources are not scaling out due to your DynamoDB service quota limits, you receive a message similar to the following. If you receive this message, it should be treated as an alarm to take action.

```
  Hello,

  A scaling action has attempted to scale out your DynamoDB resources in the
  eu-west-1 region. This operation has been prevented because it would have exceeded a
  table-level write throughput limit (Provisioned mode). This limit restricts the
  provisioned write capacity of the table and all of its associated global secondary
  indexes. To address the issue, refer to the Amazon DynamoDB Developer Guide for
  current limits and how to request higher limits [1].

  To identify your DynamoDB resources that are impacted, use the
  describe-scaling-activities command or the DescribeScalingActivities operation [2][3].
  Look for a scaling activity with StatusCode "Failed" and a StatusMessage similar to
  "Failed to set write capacity units to 45000. Reason: The requested WriteCapacityUnits,
  45000, is above the per table maximum for the account in eu-west-1. Per table maximum:
  40000." You can also view these scaling activities from the Capacity tab of your tables
  in the AWS Management Console for DynamoDB.

  We strongly recommend that you address this issue to ensure that your tables
  are prepared to handle increases in traffic. This notification is sent only once in
  each 12 hour period, even if another failed scaling action occurs.

  [1] https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Limits.html#default-
limits-throughput-capacity-modes
  [2] https://docs.aws.amazon.com/cli/latest/reference/application-autoscaling/describe-
scaling-activities.html
  [3] https://docs.aws.amazon.com/autoscaling/application/APIReference/
API_DescribeScalingActivities.html

  Sincerely,
  Amazon Web Services
```

Security in Application Auto Scaling

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Application Auto Scaling, see AWS services in scope by compliance program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Application Auto Scaling. The following topics show you how to configure Application Auto Scaling to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Application Auto Scaling resources.

Topics

Application Auto Scaling and data protection

The AWS shared responsibility model applies to data protection in Application Auto Scaling. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.

- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Application Auto Scaling or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Identity and Access Management for Application Auto Scaling

AWS Identity and Access Management (IAM) is an Amazon Web Services (AWS) service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS resources. IAM is an AWS service that you can use with no additional charge.

To use Application Auto Scaling, you need an AWS account and credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to make authenticated requests instead of using your AWS account root user credentials. You can create an IAM user and grant that user full access. We refer to these users as administrator users. You can use the administrator user credentials, instead of AWS account root user credentials, to interact with AWS and perform tasks, such as configuring scaling policies. For more information, see AWS account root user credentials vs. IAM user credentials in the *AWS General Reference* and IAM best practices in the *IAM User Guide*.

After you create an IAM user, you will need to obtain your AWS access keys if you want to access Application Auto Scaling through the Application Auto Scaling API, whether by the Query (HTTPS) interface directly or indirectly through an SDK, the AWS Command Line Interface, or the AWS Tools for Windows PowerShell. AWS access keys consist of an access key ID and a secret access key. For more information about getting your AWS access keys, see AWS security credentials in the *AWS General Reference*.

To get started quickly using Application Auto Scaling to set up automatic scaling, go through the steps in Setting up (p. 3). As you go through the example tutorial, you create an IAM user and AWS access keys for the specified user.

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Application Auto Scaling resources. For example, you must have permissions to create scaling policies, configure scheduled scaling, and so on.

The following sections provide details on how an IAM administrator can use IAM to help secure your AWS resources, by controlling who can perform Application Auto Scaling API actions.

Topics

How Application Auto Scaling works with IAM

Note
In December 2017, there was an update for Application Auto Scaling, enabling several service-linked roles for Application Auto Scaling integrated services. Specific IAM permissions *and* an Application Auto Scaling service-linked role (or a service role for Amazon EMR auto scaling) are required so that users can configure scaling.

Before you use IAM to manage access to Application Auto Scaling, you should understand what IAM features are available to use with Application Auto Scaling. To get a high-level view of how Application Auto Scaling and other AWS services work with IAM, see AWS services that work with IAM in the *IAM User Guide*.

Topics

Application Auto Scaling identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources, and the conditions under which actions are allowed or denied. Application Auto Scaling supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in Application Auto Scaling use the following prefix before the action: `application-autoscaling:`. Policy statements must include either an `Action` or `NotAction` element. Application Auto Scaling defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as shown in the following example.

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

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

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

To see a list of Application Auto Scaling API actions, see Actions in the *Application Auto Scaling API Reference*.

Resources

The `Resource` element specifies the object or objects to which the action applies.

Application Auto Scaling has no service-defined resources that can be used as the `Resource` element of an IAM policy statement. Therefore, there are no Amazon Resource Names (ARNs) for Application Auto Scaling for you to use in an IAM policy. To control access to Application Auto Scaling API actions, always use an * (asterisk) as the resource when writing an IAM policy.

Condition keys

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. For example, you might want a policy to be applied only after a specific date. To express conditions, use predefined condition keys.

Application Auto Scaling does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

The `Condition` element is optional.

Examples

To view examples of Application Auto Scaling identity-based policies, see Application Auto Scaling identity-based policy examples (p. 93).

Application Auto Scaling resource-based policies

Other AWS services, such as Amazon Simple Storage Service, support resource-based permissions policies. For example, you can attach a permissions policy to an S3 bucket to manage access permissions to that bucket.

Application Auto Scaling does not support resource-based policies.

Access Control Lists (ACLs)

Application Auto Scaling does not support Access Control Lists (ACLs).

Authorization based on Application Auto Scaling tags

Application Auto Scaling has no service-defined resources that can be tagged. Therefore, it does not support controlling access based on tags.

Application Auto Scaling IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Using temporary credentials with Application Auto Scaling

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Application Auto Scaling supports using temporary credentials.

Service-linked roles

Service-linked roles give permissions to Application Auto Scaling so that it can make specific calls to other AWS services on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Application Auto Scaling supports service-linked roles. For more information, see Service-linked roles for Application Auto Scaling (p. 90).

Service roles

If your Amazon EMR cluster uses automatic scaling, this feature allows Application Auto Scaling to assume a service role on your behalf. Similar to a service-linked role, a service role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Application Auto Scaling supports service roles only for Amazon EMR. For documentation for the EMR service role, see Using automatic scaling with a custom policy for instance groups in the *Amazon EMR Management Guide*.

Note
With the introduction of service-linked roles, several legacy service roles are no longer required. If you specify a legacy service role for any other service (for example, Amazon ECS or Spot Fleet), Application Auto Scaling ignores it. Instead, it uses a service-linked role. If this role doesn't exist, you must have permissions to create it or a permission error occurs.

AWS managed policies for Application Auto Scaling

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ViewOnlyAccess** AWS managed policy provides read-only access to many AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

Contents

AWS managed policy granting access to AppStream 2.0 and CloudWatch

Policy name: AWSApplicationAutoscalingAppStreamFleetPolicy

You can't attach `AWSApplicationAutoscalingAppStreamFleetPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon AppStream and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_AppStreamFleet` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

- Action: `appstream:DescribeFleets`
- Action: `appstream:UpdateFleet`
- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to Aurora and CloudWatch

Policy name: AWSApplicationAutoscalingRDSClusterPolicy

You can't attach `AWSApplicationAutoscalingRDSClusterPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Aurora and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_RDSCluster` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

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

AWS managed policy granting access to Amazon Comprehend and CloudWatch

Policy name: AWSApplicationAutoscalingComprehendEndpointPolicy

You can't attach `AWSApplicationAutoscalingComprehendEndpointPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon Comprehend and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

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

AWS managed policy granting access to DynamoDB and CloudWatch

Policy name: AWSApplicationAutoscalingDynamoDBTablePolicy

You can't attach `AWSApplicationAutoscalingDynamoDBTablePolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call DynamoDB and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_DynamoDBTable` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

- Action: `dynamodb:DescribeTable`
- Action: `dynamodb:UpdateTable`
- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to Amazon ECS and CloudWatch

Policy name: AWSApplicationAutoscalingECSServicePolicy

You can't attach `AWSApplicationAutoscalingECSServicePolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon ECS and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_ECSService` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

- Action: `ecs:DescribeServices`
- Action: `ecs:UpdateService`
- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to ElastiCache and CloudWatch

Policy name: AWSApplicationAutoscalingElastiCacheRGPolicy

You can't attach `AWSApplicationAutoscalingElastiCacheRGPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call ElastiCache and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on the specified resources:

- Action: `elasticache:DescribeReplicationGroups` on all resources
- Action: `elasticache:ModifyReplicationGroupShardConfiguration` on all resources
- Action: `elasticache:IncreaseReplicaCount` on all resources
- Action: `elasticache:DecreaseReplicaCount` on all resources
- Action: `elasticache:DescribeCacheClusters` on all resources
- Action: `elasticache:DescribeCacheParameters` on all resources
- Action: `cloudwatch:DescribeAlarms` on all resources

- Action: `cloudwatch:PutMetricAlarm` on the resource `arn:*:cloudwatch:*:*:alarm:TargetTracking*`
- Action: `cloudwatch:DeleteAlarms` on the resource `arn:*:cloudwatch:*:*:alarm:TargetTracking*`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to Amazon Keyspaces and CloudWatch

Policy name: AWSApplicationAutoscalingCassandraTablePolicy

You can't attach `AWSApplicationAutoscalingCassandraTablePolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon Keyspaces and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_CassandraTable` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on the specified resources:

- Action: `cassandra:Select` on the resource `arn:*:cassandra:*:*:/keyspace/system/table/*`
- Action: `cassandra:Select` on the resource `arn:*:cassandra:*:*:/keyspace/system_schema/table/*`
- Action: `cassandra:Select` on the resource `arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*`
- Action: `cassandra:Alter` on the resource `arn:*:cassandra:*:*:"*"`
- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to Lambda and CloudWatch

Policy name: AWSApplicationAutoscalingLambdaConcurrencyPolicy

You can't attach `AWSApplicationAutoscalingLambdaConcurrencyPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Lambda and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

- Action: `lambda:PutProvisionedConcurrencyConfig`
- Action: `lambda:GetProvisionedConcurrencyConfig`
- Action: `lambda:DeleteProvisionedConcurrencyConfig`

- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to Amazon MSK and CloudWatch

Policy name: AWSApplicationAutoscalingKafkaClusterPolicy

You can't attach `AWSApplicationAutoscalingKafkaClusterPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon MSK and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_KafkaCluster` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

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

AWS managed policy granting access to Neptune and CloudWatch

Policy name: AWSApplicationAutoscalingNeptuneClusterPolicy

You can't attach `AWSApplicationAutoscalingNeptuneClusterPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Neptune and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on the specified resources:

- Action: `rds:AddTagsToResource` on resources with the prefix *autoscaled-reader* in the Amazon Neptune database engine (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}`)
- Action: `rds:ListTagsForResource` on all resources
- Action: `rds:CreateDBInstance` on resources with the prefix *autoscaled-reader* in all DB clusters (`"Resource":"arn:*:rds:*:*:db:autoscaled-reader*", "arn:aws:rds:*:*:cluster:*"`) in the Amazon Neptune database engine (`"Condition":{"StringEquals":{"rds:DatabaseEngine":"neptune"}`)
- Action: `rds:DescribeDBInstances` on all resources
- Action: `rds:DescribeDBClusters` on all resources

- Action: `rds:DescribeDBClusterParameters` on all resources
- Action: `rds:DeleteDBInstance` on the resource `arn:*:rds:*:*:db:autoscaled-reader*`
- Action: `cloudwatch:DescribeAlarms` on all resources
- Action: `cloudwatch:PutMetricAlarm` on the resource `arn:*:cloudwatch:*:*:alarm:TargetTracking*`
- Action: `cloudwatch:DeleteAlarms` on the resource `arn:*:cloudwatch:*:*:alarm:TargetTracking*`
- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to SageMaker and CloudWatch

Policy name: AWSApplicationAutoscalingSageMakerEndpointPolicy

You can't attach `AWSApplicationAutoscalingSageMakerEndpointPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call SageMaker and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

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

AWS managed policy granting access to EC2 Spot Fleet and CloudWatch

Policy name: AWSApplicationAutoscalingEC2SpotFleetRequestPolicy

You can't attach `AWSApplicationAutoscalingEC2SpotFleetRequestPolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call Amazon EC2 and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

- Action: `ec2:DescribeSpotFleetRequests`
- Action: `ec2:ModifySpotFleetRequest`
- Action: `cloudwatch:DescribeAlarms`
- Action: `cloudwatch:PutMetricAlarm`

- Action: `cloudwatch:DeleteAlarms`

AWS managed policy granting access to your custom resources and CloudWatch

Policy name: AWSApplicationAutoScalingCustomResourcePolicy

You can't attach `AWSApplicationAutoScalingCustomResourcePolicy` to your AWS Identity and Access Management (IAM) entities. This policy is attached to a service-linked role that allows Application Auto Scaling to call your custom resources that are available through API Gateway and CloudWatch and perform scaling on your behalf.

Permission details

The `AWSServiceRoleForApplicationAutoScaling_CustomResource` service-linked role permissions policy allows Application Auto Scaling to complete the following actions on all related resources ("Resource": "*"):

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

Application Auto Scaling updates to AWS managed policies

View details about updates to AWS managed policies for Application Auto Scaling since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Application Auto Scaling Document history page.

Change	Description	Date
Application Auto Scaling adds Neptune policy	Application Auto Scaling added a new managed policy for Neptune. This policy is attached to a service-linked role that allows Application Auto Scaling to call Neptune and CloudWatch and perform scaling on your behalf.	October 6, 2021
Application Auto Scaling adds ElastiCache for Redis policy	Application Auto Scaling added a new managed policy for ElastiCache. This policy is attached to a service-linked role that allows Application Auto Scaling to call ElastiCache and CloudWatch and perform scaling on your behalf.	August 19, 2021
Application Auto Scaling started tracking changes	Application Auto Scaling started tracking changes for its AWS managed policies.	August 19, 2021

Service-linked roles for Application Auto Scaling

Application Auto Scaling uses service-linked roles for the permissions that it requires to call other AWS services on your behalf. A service-linked role is a unique type of AWS Identity and Access Management (IAM) role that is linked directly to an AWS service. Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role.

Contents

Overview

For services that integrate with Application Auto Scaling, Application Auto Scaling creates service-linked roles for you. There is one service-linked role for each service. Each service-linked role trusts the specified service principal to assume it. For more information, see AWS services that you can use with Application Auto Scaling (p. 15).

Application Auto Scaling includes all of the necessary permissions for each service-linked role. These managed permissions are created and managed by Application Auto Scaling, and they define the allowed actions for each resource type. For details about the permissions that each role grants, see AWS managed policies for Application Auto Scaling (p. 82).

The following sections describe how to create and manage Application Auto Scaling service-linked roles. Start by configuring permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role.

Permissions required to create a service-linked role

Application Auto Scaling requires permissions to create a service-linked role the first time any user in your AWS account calls `RegisterScalableTarget` for a given service. Application Auto Scaling creates a service-linked role for the target service in your account, if the role does not exist already. The service-linked role grants permissions to Application Auto Scaling so that it can call the target service on your behalf.

For automatic role creation to succeed, users must have permission for the `iam:CreateServiceLinkedRole` action.

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

The following is a permissions policy that allows an IAM user or role to create a service-linked role for Spot Fleet. You can specify the service-linked role in the policy's `Resource` field as an ARN, and the service principal for your service-linked role as a condition, as shown. For the ARN for each service, see Service-linked role ARN reference (p. 92).

```
{
    "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

The `iam:AWSServiceName` IAM condition key specifies the service principal to which the role is attached, which is indicated in this example policy as *`ec2.application-autoscaling`*`.amazonaws.com`. Do not try to guess the service principal. To view the service principal for a service, see AWS services that you can use with Application Auto Scaling (p. 15).

Create service-linked roles (automatic)

You don't need to manually create a service-linked role. Application Auto Scaling creates the appropriate service-linked role for you when you call `RegisterScalableTarget`. For example, if you set up automatic scaling for an Amazon ECS service, Application Auto Scaling creates the `AWSServiceRoleForApplicationAutoScaling_ECSService` role.

Create service-linked roles (manual)

To create the service-linked role, you can use the IAM console, AWS CLI, or IAM API. For more information, see Creating a service-linked role in the *IAM User Guide*.

To create a service-linked role (AWS CLI)

Use the following create-service-linked-role CLI command to create the Application Auto Scaling service-linked role. In the request, specify the service name "prefix".

To find the service name prefix, refer to the information about the service principal for the service-linked role for each service in the AWS services that you can use with Application Auto Scaling (p. 15) section. The service name and the service principal share the same prefix. For example, to create the AWS Lambda service-linked role, use `lambda.application-autoscaling.amazonaws.com`.

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

Edit the service-linked roles

With the service-linked roles created by Application Auto Scaling, you can edit only their descriptions. For more information, see Editing a service-linked role in the *IAM User Guide*.

Delete the service-linked roles

If you no longer use Application Auto Scaling with a supported service, we recommend that you delete the corresponding service-linked role.

You can delete a service-linked role only after first deleting the related AWS resources. This protects you from inadvertently revoking Application Auto Scaling permissions to your resources. For more information, see the documentation for the scalable resource. For example, to delete an Amazon ECS service, see Deleting a service in the *Amazon Elastic Container Service Developer Guide*.

You can use IAM to delete a service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

After you delete a service-linked role, Application Auto Scaling creates the role again when you call `RegisterScalableTarget`.

Supported Regions for Application Auto Scaling service-linked roles

Application Auto Scaling supports using service-linked roles in all of the AWS Regions where the service is available.

Service-linked role ARN reference

Service	ARN
AppStream 2.0	`arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`
Aurora	`arn:aws:iam::012345678910:role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster`
Comprehend	`arn:aws:iam::012345678910:role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint`
DynamoDB	`arn:aws:iam::012345678910:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`
ECS	`arn:aws:iam::012345678910:role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService`
ElastiCache	`arn:aws:iam::012345678910:role/aws-service-role/elasticache.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`
Keyspaces	`arn:aws:iam::012345678910:role/aws-service-role/cassandra.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable`
Lambda	`arn:aws:iam::012345678910:role/aws-service-role/lambda.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency`
MSK	`arn:aws:iam::012345678910:role/aws-service-role/kafka.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_KafkaCluster`
Neptune	`arn:aws:iam::012345678910:role/aws-service-role/neptune.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`

Service	ARN
SageMaker	`arn:aws:iam::012345678910:role/aws-service-role/sagemaker.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`
Spot Fleets	`arn:aws:iam::012345678910:role/aws-service-role/ec2.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`
Custom resources	`arn:aws:iam::012345678910:role/aws-service-role/custom-resource.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CustomResource`

Note
You can specify the ARN of a service-linked role for the `RoleARN` property of an AWS::ApplicationAutoScaling::ScalableTarget resource in your AWS CloudFormation stack templates, even if the specified service-linked role doesn't yet exist. Application Auto Scaling automatically creates the role for you.

Application Auto Scaling identity-based policy examples

By default, a brand new IAM user has no permissions to do anything. An IAM administrator must create IAM policies that grant users and roles permission to perform Application Auto Scaling API actions, such as configuring scaling policies. The administrator must then attach those policies to the IAM users or roles that require the permissions.

To learn how to create an IAM policy using the following example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Contents

Permissions required for Application Auto Scaling API actions

The following policies grant permissions for common use cases when calling Application Auto Scaling API. Refer to this section when setting up Access control (p. 79) and writing permissions policies that you can attach to an IAM user or role. Each policy grants access to all or some of the Application Auto Scaling API actions. You also need to make sure that the IAM user or role has a permissions policy for the target service and CloudWatch (see the next section for details).

The following permissions policy grants access to all Application Auto Scaling API actions.

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

The following permissions policy grants access to all Application Auto Scaling API actions that are required to configure scaling policies and not scheduled actions.

```
{
    "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": "*"
        }
    ]
}
```

The following permissions policy grants access to all Application Auto Scaling API actions that are required to configure scheduled actions and not scaling policies.

```
{
    "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": "*"
        }
    ]
}
```

Permissions required for API actions on target services and CloudWatch

To successfully configure and use Application Auto Scaling with the target service, IAM users must be granted the required permissions for Amazon CloudWatch and for each target service for which they will configure scaling. Use the following policies to give users the minimum permissions required to work with target services and CloudWatch.

Contents

AppStream 2.0 fleets

The following permissions policy grants access to all AppStream 2.0 and CloudWatch API actions that are required.

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

Aurora replicas

The following permissions policy grants access to all Aurora and CloudWatch API actions that are required.

```
{
    "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 document classification and entity recognizer endpoints

The following permissions policy grants access to all Amazon Comprehend and CloudWatch API actions that are required.

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

DynamoDB tables and global secondary indexes

The following permissions policy grants access to all DynamoDB and CloudWatch API actions that are required.

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

ECS services

The following permissions policy grants access to all ECS and CloudWatch API actions that are required.

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

ElastiCache replication groups

The following permissions policy grants access to all ElastiCache and CloudWatch API actions that are required.

```
{
    "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 clusters

The following permissions policy grants access to all Amazon EMR and CloudWatch API actions that are required.

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

Amazon Keyspaces tables

The following permissions policy grants access to all Amazon Keyspaces and CloudWatch API actions that are required.

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

Lambda functions

The following permissions policy grants access to all Lambda and CloudWatch API actions that are required.

```
{
    "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) broker storage

The following permissions policy grants access to all Amazon MSK and CloudWatch API actions that are required.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "kafka:DescribeCluster",
              "kafka:DescribeClusterOperation",
              "kafka:UpdateBrokerStorage",
              "cloudwatch:DescribeAlarms",
              "cloudwatch:PutMetricAlarm",
              "cloudwatch:DeleteAlarms"
            ],
            "Resource": "*"
        }
    ]
}
```

Neptune clusters

The following permissions policy grants access to all Neptune and CloudWatch API actions that are required.

```
{
    "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 endpoints

The following permissions policy grants access to all SageMaker and CloudWatch API actions that are required.

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

Spot Fleets (Amazon EC2)

The following permissions policy grants access to all Spot Fleet and CloudWatch API actions that are required.

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

Custom resources

The following permissions policy grants a user the required permission for the API Gateway API executing action. This policy also grants access to all CloudWatch actions that are required.

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

Permissions for working in the AWS Management Console

There is no standalone Application Auto Scaling console. Most services that integrate with Application Auto Scaling have features that are dedicated to helping you configure scaling with their console.

In most cases, each service provides AWS managed (predefined) IAM policies that define access to their console, which includes permissions to the Application Auto Scaling API actions. For more information, refer to the documentation for the service whose console you want to use.

You can also create your own custom IAM policies to give users fine-grained permissions to view and work with specific Application Auto Scaling API actions in the AWS Management Console. You can use the example policies in the previous sections; however, they are designed for requests that are made with the AWS CLI or an SDK. The console uses additional API actions for its features, so these policies may not work as expected. For example, to configure step scaling, users might require additional permissions to create and manage CloudWatch alarms.

> **Tip**
> To help you work out which API actions are required to perform tasks in the console, you can use a service such as AWS CloudTrail. For more information, see the AWS CloudTrail User Guide.

The following shows an example of a permissions policy that allows a user to configure scaling policies for Spot Fleet. In addition to the IAM permissions for Spot Fleet, the IAM user that accesses fleet scaling settings from the console must have the appropriate permissions for the services that support dynamic scaling.

```
{
    "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"
                }
            }
        }
    ]
}
```

This policy allows users to view and modify scaling policies in the Amazon EC2 console, and to create and manage CloudWatch alarms in the CloudWatch console.

You can adjust the API actions to limit user access. For example, replacing `application-autoscaling:*` with `application-autoscaling:Describe*` means that the user has read-only access.

You can also adjust the CloudWatch permissions as required to limit user access to CloudWatch features. For more information, see Permissions required to use the CloudWatch console in the *Amazon CloudWatch User Guide*.

Troubleshooting access to Application Auto Scaling

If you encounter `AccessDeniedException` or similar difficulties when working with Application Auto Scaling, consult the information in this section.

I am not authorized to perform an action in Application Auto Scaling

If you receive an `AccessDeniedException` when calling an AWS API operation, it means that the AWS Identity and Access Management (IAM) user or role credentials that you are using do not have the required permissions to make that call.

The following example error occurs when the `mateojackson` IAM user tries to view details about a scalable target, but does not have `application-autoscaling:DescribeScalableTargets` permission.

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

If you receive this or similar errors, then you must contact your administrator for assistance.

An administrator for your account will need to make sure that your IAM user or role has permissions to access all of the API actions that Application Auto Scaling uses to access resources in the target service and CloudWatch. There are different permissions required depending on which resources you are working with. Application Auto Scaling also requires permission to create a service-linked role the first time that a user configures scaling for a given resource.

I'm an administrator and want to allow others to access Application Auto Scaling

To allow others to access Application Auto Scaling, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Application Auto Scaling.

To get started, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I'm an administrator and my IAM policy returned an error or isn't working as expected

In addition to the IAM permissions that are required for Application Auto Scaling API actions, your IAM permissions policies must grant access to call the target service and CloudWatch.

If a user or application doesn't have the proper IAM policy permissions, their access might be unexpectedly denied. To write permissions policies for users and applications in your accounts, consult the information in Application Auto Scaling identity-based policy examples (p. 93).

For information about how validation is performed, see Permissions validation for API calls on target resources (p. 102).

Note that some permission issues can also be due to an issue with creating the service-linked roles used by Application Auto Scaling. For information about creating these service-linked roles, see Service-linked roles for Application Auto Scaling (p. 90).

Permissions validation for API calls on target resources

Making authorized requests to Application Auto Scaling API actions requires that the API caller has permissions to access AWS resources in the target service and in CloudWatch. Application Auto Scaling validates permissions for requests associated with both the target service and CloudWatch before proceeding with the request. To accomplish this, we issue a series of calls to validate the IAM permissions on target resources. When a response is returned, it is read by Application Auto Scaling. If the IAM permissions do not allow a given action, Application Auto Scaling fails the request and returns an error to the user containing information about the missing permission. This ensures that the scaling configuration that the user wants to deploy functions as intended, and that a useful error is returned if the request fails.

As an example of how this works, the following information provides details about how Application Auto Scaling performs permissions validations with Aurora and CloudWatch.

When an IAM user calls the `RegisterScalableTarget` API against an Aurora DB cluster, Application Auto Scaling performs all of the following checks to verify that the IAM user has the required permissions (in bold).

- **rds:CreateDBInstance**: To determine whether the user has this permission, we send a request to the `CreateDBInstance` API operation, attempting to create a DB instance with invalid parameters (empty instance ID) in the Aurora DB cluster that the user specified. For an authorized user, the API returns an `InvalidParameterValue` error code response after it audits the request. However, for an unauthorized user, we get an `AccessDenied` error and fail the Application Auto Scaling request with a `ValidationException` error to the user that lists the missing permissions.
- **rds:DeleteDBInstance**: We send an empty instance ID to the `DeleteDBInstance` API operation. For an authorized user, this request results in an `InvalidParameterValue` error. For an unauthorized user, it results in `AccessDenied` and sends a validation exception to the user (same treatment as described in the first bullet point).
- **rds:AddTagsToResource**: Because the `AddTagsToResource` API operation requires an Amazon Resource Name (ARN), it is necessary to specify a "dummy" resource using an invalid account ID (12345) and dummy instance ID (non-existing-db) to construct the ARN (`arn:aws:rds:us-east-1:12345:db:non-existing-db`). For an authorized user, this request results in an `InvalidParameterValue` error. For an unauthorized user, it results in `AccessDenied` and sends a validation exception to the user.
- **rds:DescribeDBCluster**: We describe the cluster name for the resource being registered for auto scaling. For an authorized user, we get a valid describe result. For an unauthorized user, it results in `AccessDenied` and sends a validation exception to the user.
- **rds:DescribeDBInstance**. We call the `DescribeDBInstance` API with a `db-cluster-id` filter that filters on the cluster name that was provided by the user to register the scalable target. For an authorized user, we are permitted to describe all of the DB instances in the DB cluster. For an unauthorized user, this call results in `AccessDenied` and sends a validation exception to the user.
- **cloudwatch:PutMetricAlarm**: We call the `PutMetricAlarm` API without any parameters. Because alarm name is missing, the request results in `ValidationError` for an authorized user. For an unauthorized user, it results in `AccessDenied` and sends a validation exception to the user.
- **cloudwatch:DescribeAlarms**: We call the `DescribeAlarms` API with the maximum number of records value set to 1. For an authorized user, we expect information on one alarm in the response. For an unauthorized user, this call results in `AccessDenied` and sends a validation exception to the user.
- **cloudwatch:DeleteAlarms**: Similar to `PutMetricAlarm` above, we provide no parameters to `DeleteAlarms` request. Because an alarm name is missing from the request, this call fails with `ValidationError` for an authorized user. For an unauthorized user, it results in `AccessDenied` and sends a validation exception to the user.

Whenever any one of these validation exceptions occur, it is logged. You can take steps to manually identify which calls failed validation by using AWS CloudTrail. For more information, see the AWS CloudTrail User Guide.

Compliance validation for Application Auto Scaling

The security and compliance of Amazon Web Services (AWS) services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using Application Auto Scaling is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and compliance quick start guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA security and compliance whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating resources with rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Application Auto Scaling

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

Infrastructure security in Application Auto Scaling

As a managed service, Application Auto Scaling is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

You use AWS published API calls to access Application Auto Scaling through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Application Auto Scaling and interface VPC endpoints

You can establish a private connection between your virtual private cloud (VPC) and the Application Auto Scaling API by creating an interface VPC endpoint. You can use this connection to call the Application

Auto Scaling API from your VPC without sending traffic over the internet. The endpoint provides reliable, scalable connectivity to the Application Auto Scaling API. It does this without requiring an internet gateway, NAT instance, or VPN connection.

Interface VPC endpoints are powered by AWS PrivateLink, a feature that enables private communication between AWS services using private IP addresses. For more information, see AWS PrivateLink.

Note
You must explicitly enable each API that you want to access through an interface VPC endpoint. For example, you might need to also configure an interface VPC endpoint for `autoscaling.region.amazonaws.com` if you're using the Amazon EC2 Auto Scaling API operations. For more information, see Amazon EC2 Auto Scaling and interface VPC endpoints in the *Amazon EC2 Auto Scaling User Guide*.

Create an interface VPC endpoint

You can create a VPC endpoint for the Application Auto Scaling service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). Create an endpoint for Application Auto Scaling using the following service name:

- **com.amazonaws.*region*.application-autoscaling** — Creates an endpoint for the Application Auto Scaling API operations.

- **cn.com.amazonaws.*region*.application-autoscaling** — Creates an endpoint for the Application Auto Scaling API operations in the Amazon Web Services China (Beijing) Region and Amazon Web Services China (Ningxia) Region.

For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Enable private DNS for the endpoint to make API requests to the supported service using its default DNS hostname (for example, `application-autoscaling.us-east-1.amazonaws.com`). When creating an endpoint for AWS services, this setting is enabled by default. For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

You do not need to change any Application Auto Scaling settings. Application Auto Scaling calls other AWS services using either service endpoints or private interface VPC endpoints, whichever are in use.

Create a VPC endpoint policy

You can attach a policy to your VPC endpoint to control access to the Application Auto Scaling API. The policy specifies:

- The principal that can perform actions.
- The actions that can be performed.
- The resource on which the actions can be performed.

The following example shows a VPC endpoint policy that denies everyone permission to delete a scaling policy through the endpoint. The example policy also grants everyone permission to perform all other actions.

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

For more information, see Using VPC endpoint policies in the *Amazon VPC User Guide*.

Endpoint migration

Application Auto Scaling recently introduced `application-autoscaling.region.amazonaws.com` as the new default DNS hostname and endpoint for calls to the Application Auto Scaling API. The new endpoint is compatible with the latest release of the AWS CLI and SDKs. If you have not done so already, install the latest AWS CLI and SDKs to use the new endpoint. To update the AWS CLI, see Installing the AWS CLI using pip in the *AWS Command Line Interface User Guide*. For information about the AWS SDKs, see Tools for Amazon Web Services.

> **Note**
> For backward compatibility, the `autoscaling.region.amazonaws.com` endpoint will continue to be supported for calls to the Application Auto Scaling API. To set up the `autoscaling.region.amazonaws.com` endpoint as a private interface VPC endpoint, see Amazon EC2 Auto Scaling and interface VPC endpoints in the *Amazon EC2 Auto Scaling User Guide*.

Endpoint to Call When Using the CLI or the AWS API

For the current release of Application Auto Scaling, your calls to the Application Auto Scaling API automatically go to the `application-autoscaling.region.amazonaws.com` endpoint instead of `autoscaling.region.amazonaws.com`.

You can call the new endpoint in the CLI by using the following parameter with each command to specify the endpoint: `--endpoint-url https://application-autoscaling.region.amazonaws.com`.

Although it is not recommended, you can also call the old endpoint in the CLI by using the following parameter with each command to specify the endpoint: `--endpoint-url https://autoscaling.region.amazonaws.com`.

For the various SDKs used to call the APIs, see the documentation for the SDK of interest to learn how to direct the requests to a specific endpoint. For more information, see Tools for Amazon Web Services.

Application Auto Scaling service quotas

Your AWS account has the following default quotas, formerly referred to as limits, for Application Auto Scaling.

To request an increase, use the Application Auto Scaling limits form. Make sure that you specify the type of resource with your request for an increase, for example, Amazon ECS or DynamoDB.

Default quotas per Region per account

Item	Default
Maximum number of scalable targets per resource type	Quotas vary depending on resource type. Up to 3000 Amazon DynamoDB and ECS scalable targets and 500 scalable targets each for all other resource types.
Maximum number of scaling policies per scalable target	50 Includes both step scaling policies and target tracking policies.
Maximum number of scheduled actions per scalable target	200
Maximum number of step adjustments per step scaling policy	20

Keep service quotas in mind as you scale out your workloads. For example, when you reach the maximum number of capacity units allowed by a service, scaling out will stop. If demand drops and the current capacity decreases, Application Auto Scaling can scale out again. To avoid reaching this service quota limit again, you can request an increase. Each service has its own default quotas for the maximum capacity of the resource. For information about the default quotas for other AWS services, see Service endpoints and quotas in the *Amazon Web Services General Reference*.

Creating Application Auto Scaling resources with AWS CloudFormation

Application Auto Scaling is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want, and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your Application Auto Scaling resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Application Auto Scaling and AWS CloudFormation templates

To provision and configure resources for Application Auto Scaling and related services, you must understand AWS CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see What is AWS CloudFormation Designer? in the *AWS CloudFormation User Guide*.

Application Auto Scaling supports creating scaling policies and scheduled actions in AWS CloudFormation. For more information, see the Application Auto Scaling reference in the *AWS CloudFormation User Guide*.

Example template snippets

We also provide a few JSON and YAML template snippets that you can use to understand how to declare various scaling policies and scheduled actions in your stack templates. For more information, see the Application Auto Scaling template examples page and the AWS::ApplicationAutoScaling::ScalingPolicy reference in the *AWS CloudFormation User Guide*.

When you create a stack template for Application Auto Scaling resources, you must provide the following:

- A namespace for the target service (for example, **`appstream`**). See the AWS::ApplicationAutoScaling::ScalableTarget reference to obtain service namespaces.
- A scalable dimension associated with the target resource (for example, **`appstream:fleet:DesiredCapacity`**). See the AWS::ApplicationAutoScaling::ScalableTarget reference to obtain scalable dimensions.
- A resource ID for the target resource (for example, **`fleet/sample-fleet`**). See the AWS::ApplicationAutoScaling::ScalableTarget reference for information about the syntax and examples of specific resource IDs.
- A service-linked role for the target resource (for example, **`arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/`**

`AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`). See the Service-linked role ARN reference (p. 92) table to obtain role ARNs.

You can create stack templates that create scaling policies for any of the resources that Application Auto Scaling can scale. However, the types of scaling policies that the services support can differ. To find out which type of scaling policies are not supported for your service, see the `PolicyType` property in the AWS::ApplicationAutoScaling::ScalingPolicy reference.

Learn more about AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation User Guide
- AWS CloudFormation API Reference
- AWS CloudFormation Command Line Interface User Guide

Document history

The following table describes important additions to the Application Auto Scaling documentation, beginning in January 2018. For notification about updates to this documentation, you can subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Application Auto Scaling now reports changes to its AWS managed policies (p. 110)	Beginning August 19, 2021, changes to managed policies are reported in the topic Application Auto Scaling updates to AWS managed policies. The first change listed is the addition of permissions needed for ElastiCache for Redis.	August 19, 2021
Add support for ElastiCache for Redis replication groups (p. 110)	Use Application Auto Scaling to scale the number of node groups and the number of replicas per node group for an ElastiCache for Redis replication group (cluster). For more information, see ElastiCache for Redis and Application Auto Scaling.	August 19, 2021
Guide changes (p. 110)	New IAM topics in the *Application Auto Scaling User Guide* help you troubleshoot access to Application Auto Scaling. For more information, see Identity and Access Management for Application Auto Scaling. Also added new example IAM permissions policies for actions on target services and Amazon CloudWatch. For more information, see Example policies for working with the AWS CLI or an SDK.	February 23, 2021
Add support for local time zones (p. 110)	You can now create scheduled actions in the local time zone. If your time zone observes daylight saving time, it automatically adjusts for Daylight Saving Time (DST). For more information, see Scheduled scaling.	February 2, 2021
Guide changes (p. 110)	A new tutorial in the *Application Auto Scaling User Guide* helps you understand how to use target tracking scaling policies	October 15, 2020

	and scheduled scaling to increase the availability of your application when using Application Auto Scaling. Also, a new topic explains how to trigger a notification when CloudWatch has detected any problems that might require your attention.	
Add support for Amazon Managed Streaming for Apache Kafka cluster storage (p. 110)	Use a target tracking scaling policy to scale out the amount of broker storage associated with an Amazon MSK cluster.	September 30, 2020
Add support for Amazon Comprehend entity recognizer endpoints (p. 110)	Use Application Auto Scaling to scale the number of inference units provisioned for your Amazon Comprehend entity recognizer endpoints.	September 28, 2020
Add support for Amazon Keyspaces (for Apache Cassandra) tables (p. 110)	Use Application Auto Scaling to scale the provisioned throughput (read and write capacity) of an Amazon Keyspaces table.	April 23, 2020
New "Security" chapter (p. 110)	A new Security chapter in the *Application Auto Scaling User Guide* helps you understand how to apply the shared responsibility model when using Application Auto Scaling. As part of this update, the user guide chapter "Authentication and Access Control" has been replaced by a new, more useful section, Identity and Access Management for Application Auto Scaling.	January 16, 2020
Minor updates (p. 110)	Various improvements and corrections.	January 15, 2020
Add notification functionality (p. 110)	Application Auto Scaling now sends events to Amazon EventBridge and notifications to your AWS Personal Health Dashboard when certain actions occur. For more information, see Application Auto Scaling monitoring.	December 20, 2019
Add support for AWS Lambda functions (p. 110)	Use Application Auto Scaling to scale the provisioned concurrency of a Lambda function.	December 3, 2019

Add support for Amazon Comprehend document classification endpoints (p. 110)	Use Application Auto Scaling to scale the throughput capacity of an Amazon Comprehend document classification endpoint.	November 25, 2019
Add AppStream 2.0 support for target tracking scaling policies (p. 110)	Use target tracking scaling policies to scale the size of an AppStream 2.0 fleet.	November 25, 2019
Support for Amazon VPC endpoints (p. 110)	You can now establish a private connection between your VPC and Application Auto Scaling. For migration considerations and instructions, see Application Auto Scaling and interface VPC endpoints.	November 22, 2019
Suspend and resume scaling (p. 110)	Added support for suspending and resuming scaling. For more information, see Suspending and resuming scaling for Application Auto Scaling.	August 29, 2019
New section (p. 110)	The Setting up section has been added to the Application Auto Scaling documentation. Minor improvements and fixes have been made throughout the user guide.	June 28, 2019
Guide changes (p. 110)	Improved Application Auto Scaling documentation in the Scheduled scaling, Step scaling policies, and Target tracking scaling policies sections.	March 11, 2019
Add support for custom resources (p. 110)	Use Application Auto Scaling to scale custom resources provided by your own applications or services. For more information, see our GitHub repository.	July 9, 2018
Add support for SageMaker endpoint variants (p. 110)	Use Application Auto Scaling to scale the number of endpoint instances provisioned for a variant.	February 28, 2018

The following table describes important changes to the Application Auto Scaling documentation before January 2018.

Change	Description	Date
Add support for Aurora Replicas	Use Application Auto Scaling to scale the desired count. For more information, see Using Amazon Aurora Auto Scaling	November 17, 2017

Change	Description	Date
	with Aurora replicas in the *Amazon RDS User Guide*.	
Add support for scheduled scaling	Use scheduled scaling to scale resources at specific preset times or intervals. For more information, see Scheduled scaling for Application Auto Scaling.	November 8, 2017
Add support for target tracking scaling policies	Use target tracking scaling policies to set up dynamic scaling for your application in just a few simple steps. For more information, see Target tracking scaling policies for Application Auto Scaling.	July 12, 2017
Add support for provisioned read and write capacity for DynamoDB tables and global secondary indexes	Use Application Auto Scaling to scale provisioned throughput (read and write capacity). For more information, see Managing throughput capacity with DynamoDB Auto Scaling in the *Amazon DynamoDB Developer Guide*.	June 14, 2017
Add support for AppStream 2.0 fleets	Use Application Auto Scaling to scale the size of the fleet. For more information, see Fleet Auto Scaling for AppStream 2.0 in the *Amazon AppStream 2.0 Administration Guide*.	March 23, 2017
Add support for Amazon EMR clusters	Use Application Auto Scaling to scale the core and task nodes. For more information, see Using automatic scaling in Amazon EMR in the *Amazon EMR Management Guide*.	November 18, 2016
Add support for Spot Fleets	Use Application Auto Scaling to scale the target capacity. For more information, see Automatic scaling for Spot fleet in the *Amazon EC2 User Guide for Linux Instances*.	September 1, 2016
Add support for Amazon ECS services	Use Application Auto Scaling to scale the desired count. For more information, see Service Auto Scaling in the *Amazon Elastic Container Service Developer Guide*.	August 9, 2016