Application Auto Scaling

User Guide

Application Auto Scaling: User Guide

The AWS Documentation website is getting a new look!

Try it now and let us know what you think. Switch to the new look >>

You can return to the original look by selecting English in the language selector above.

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:

- Amazon ECS services
- Spot Fleet requests
- Amazon EMR clusters
- AppStream 2.0 fleets
- DynamoDB tables and global secondary indexes
- Aurora replicas
- Amazon SageMaker endpoint variants
- Custom resources provided by your own applications or services. For more information, see the GitHub repository.

You have several options for scaling with AWS. For information about scaling your fleet of Amazon EC2 instances, see the Amazon EC2 Auto Scaling User Guide.

You can also use Application Auto Scaling and Amazon EC2 Auto Scaling in combination with AWS Auto Scaling to scale resources across multiple services. AWS Auto Scaling can help you maintain optimal availability and performance by combining predictive scaling and dynamic scaling (proactive and reactive approaches, respectively) together to scale your Amazon EC2 capacity faster. 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.

Getting Started

Application Auto Scaling integrates with all of the following services, so that you can call API actions directly from the console of the resources that you want to scale. If you are a first-time user of Application Auto Scaling, we recommend that you refer to the following documentation for the services you're interested in to learn more about how they integrate with Application Auto Scaling. These topics contain information that's especially helpful for users who mainly interact with Application Auto Scaling by using the AWS Management Console.

- Service Auto Scaling in the *Amazon Elastic Container Service Developer Guide*
- Automatic Scaling for Spot Fleet in the *Amazon EC2 User Guide*
- Using Automatic Scaling in Amazon EMR in the *Amazon EMR Management Guide*

- Fleet Auto Scaling for AppStream 2.0 in the *Amazon AppStream 2.0 Administration Guide*
- Managing Throughput Capacity with DynamoDB Auto Scaling in the *Amazon DynamoDB Developer Guide*
- Using Amazon Aurora Auto Scaling with Aurora Replicas in the *Amazon RDS User Guide*
- Automatically Scaling Amazon SageMaker Models in the *Amazon SageMaker Developer Guide*

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

If you would like the Application Auto Scaling API actions to be specified using command line options, you also have option of using this user guide. To get started, complete the exercises in Getting Started Using the AWS CLI (p. 5). In this tutorial, we show you how to use the AWS Command Line Interface (AWS CLI) to programmatically access Application Auto Scaling. However, if, at any time, you need information not included in this user guide, including examples of scaling policies that you can try, see the above service documentation.

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 services listed in the Getting Started section.

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*.

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.

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.

For information about your credentials for accessing AWS, see AWS Security Credentials in the *Amazon Web Services General Reference*. For information about regions and endpoints for Application Auto Scaling, see AWS Regions and Endpoints in the *AWS General Reference*.

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 AWS

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, 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 AWS

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 AWS Regions and Endpoints 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 AWS (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 AWS 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 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 Authentication and Access Control for Application Auto Scaling (p. 29).

The CLI commands in this tutorial were tested on Linux. To use the samples with Microsoft Windows, change the line breaks from backslashes (\) to carets (^). For information about using the CLI commands on Windows, see Specifying Parameter Values for the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

> **Note**
> You may incur AWS charges as part of this tutorial. Please monitor your Free Tier usage and make sure that you understand the AWS charges involved.

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, 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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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 example in Scheduled Scaling for Application Auto Scaling (p. 23).
- 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. 12).

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.

```
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.

```
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.

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

```
--profile adminuser
```

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.

You can have one or more target tracking scaling policies, one or more step scaling policies, or both. 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.

Limits

- You cannot create target tracking scaling policies for Amazon EMR clusters or AppStream 2.0 fleets.
- You cannot use Application Auto Scaling to create or update a scaling policy that's used with the AWS Auto Scaling service. For information about the AWS Auto Scaling console, CLI, or API actions, see the AWS Auto Scaling documentation.
- You can use Application Auto Scaling to apply a target tracking scaling policy based on a predefined or CloudWatch customized metric. Not all services allow you to manage customized metrics through the console, however. To see if a service supports customized metrics in the console, consult the documentation for that service.

Considerations

Keep the following considerations in mind:

- Not all metrics work for target tracking. This can be important when you are specifying a customized 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.
- Wherever possible, you should scale on metrics with a 1-minute frequency because that ensures a faster response to utilization changes.
- 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.
- A target tracking scaling policy does not perform scaling when the specified metric has insufficient data, for example, because of a network connectivity issue. It does not perform scale in because it does not interpret insufficient data as low utilization.
- 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.
- 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 different scaling policy type for scale in while using a target tracking scaling policy for scale out.
- 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 deleted automatically when you delete the scaling policies.

Cooldown Period

The *scale-out cooldown period* is the amount of time, in seconds, after a scale-out activity completes before another scale-out activity can start. While this cooldown period is in effect, the capacity added by the initiating scale-out event is calculated as part of the desired capacity for the next scale-out event. The intention is to continuously (but not excessively) scale out.

The *scale-in cooldown period* is the amount of time, in seconds, after a scale-in activity completes before another scale-in activity can start. This cooldown period is used to block subsequent scale-in events until it has expired. The intention is to scale in conservatively to protect your application's availability. However, if another alarm triggers a scale-out policy during the cooldown period after a scale-in event, Application Auto Scaling scales out your scalable target immediately.

Register Scalable Target

Before you can create a scaling policy, you must register the scalable target. Use the **register-scalable-target** command to register a new 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.

```
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
When you configure scaling policies in the console, this automatically registers the resource as a scalable target with Application Auto Scaling. For more information, see the documentation in the Getting Started (p. 1) section.

Create a Target Tracking Scaling Policy

You can create target tracking scaling policies that tell Application Auto Scaling to scale out or scale in automatically when the load on the application changes.

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"
    }
}
```

Alternatively, you can customize the metric used 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"
   }
}
```

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`.

```
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
```

The following is example output. It contains 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. This syntax for `query` works on Linux or macOS. On Windows, change the single quotes to double quotes.

```
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": [
            {
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmHigh-d4f0770c-b46e-434a-a60f-3b36d653feca",
                "AlarmARN": "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-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
                "AlarmARN": "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.

```
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.

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.

After a scaling activity is started, the policy continues to respond to additional alarms, even while a scaling activity is in progress. Therefore, all alarms that are breached are evaluated by Application Auto Scaling as it receives the alarm messages.

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. 12). 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.

Limits

- You cannot create step scaling policies for DynamoDB tables and global secondary indexes.

Scaling Adjustment Types

When a step scaling policy is performed, it changes the current capacity of your scalable target using the scaling adjustment specified in the policy. A scaling adjustment can't change the capacity of the scalable target above the maximum capacity or below the minimum capacity.

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.

  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.

  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. 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`.

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.

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.

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

There are a few rules for the step adjustments for your policy:

- 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.

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, suppose that you have an alarm with a breach threshold of 50 and a scaling adjustment of type `PercentChangeInCapacity`. You also have scale-out and scale-in policies with the following step adjustments:

Scale out policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
0	10	0	50 <= *value* < 60
10	20	10	60 <= *value* < 70
20	null	30	70 <= *value* < +infinity

Scale in policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
-10	0	0	40 < *value* <= 50
-20	-10	-10	30 < *value* <= 40
null	-20	-30	-infinity < *value* <= 30

Your scalable target has both a current capacity and a desired capacity of 10. 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 group 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).

Cooldown Period

The *cooldown period* is the amount of time, in seconds, after a scaling activity completes where previous trigger-related scaling activities can influence future scaling events.

While the cooldown period is in effect, capacity added by the initiating scale-out event is calculated as part of the desired capacity for the next scale-out event. The intention is to continuously (but not excessively) scale out. 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.

For scale in policies, the cooldown period is used to block subsequent scale in events until it has expired. The intention is to scale in conservatively to protect your application's availability. However, if another alarm triggers a scale-out policy during the cooldown period after a scale in event, Application Auto Scaling scales out your scalable target immediately.

Register Scalable Target

Before you can create a scaling policy, you must register the scalable target. Use the **register-scalable-target** command to register a new 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.

```
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
When you configure scaling policies in the console, this automatically registers the resource as a scalable target with Application Auto Scaling. For more information, see the documentation in the Getting Started (p. 1) section.

Configure Step Scaling Policies Using the AWS CLI

You can create step scaling policies that tell Application Auto Scaling what to do when the load on the application changes.

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%):

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

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`:

```
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"
}
```

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% 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`.

```
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. This syntax for `query` works on Linux or macOS. On Windows, change the single quotes to double quotes.

```
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.

```
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 for predictable load changes. For example, 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 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, and the minimum and maximum capacity. At the specified time, Application Auto Scaling scales based on the new capacity values.

Application Auto Scaling guarantees the order of execution for scheduled actions for the same scalable target but not for scheduled actions across scalable targets.

> **Note**
> For brevity, the examples in this topic illustrate CLI commands for Amazon ECS, DynamoDB, AppStream 2.0, Spot Fleet, and a custom resource. 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`.

Register Scalable Target

Before you can create a scheduled action, you must register the scalable target. Use the **register-scalable-target** command to register a new 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.

```
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
```

Using the AWS CLI to Create or Update a Scheduled Action

You can create and update scheduled actions that scale one time only or that scale on a recurring schedule using the **put-scheduled-action** command. When you specify the new capacity, you can specify a minimum capacity, a maximum capacity, or both.

Schedule Actions for a One-Time Schedule

You can specify a one-time schedule to automatically scale your scalable target at a certain date and time, in UTC.

Example Example: To scale in one time only

At the date and time specified for `--schedule`, if the value specified for `MaxCapacity` is below the current capacity, Application Auto Scaling scales in to `MaxCapacity`.

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --scalable-dimension ecs:service:DesiredCount \
  --resource-id service/default/web-app \
  --scheduled-action-name my-one-time-action \
  --schedule "at(2019-01-31T17:00:00)" \
  --scalable-target-action MaxCapacity=10
```

Example Example: To scale out one time only

At the date and time specified for `--schedule`, if the value specified for `MinCapacity` is above the current capacity, Application Auto Scaling scales out to `MinCapacity`.

```
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 my-one-time-action \
  --schedule "at(2019-03-31T22:00:00)" \
  --scalable-target-action MinCapacity=3
```

The `custom-resource-id.txt` file specifies the API Gateway endpoint for your custom resources. For more information about configuring a custom resource, see our GitHub repository. The contents of the file may look something like this:

Example

```
https://example.execute-api.region.amazonaws.com/prod/scalableTargetDimensions/1-23456789
```

Schedule Actions on a Recurring Schedule

You can specify a recurrence schedule for a scheduled action using the `--schedule` option of the **put-scheduled-action** command. Application Auto Scaling supports cron and rate formats for schedule expressions. For more information, see Cron Expressions and Rate Expressions in the *Amazon CloudWatch Events User Guide*.

Example Example: To scale on a recurring schedule using a cron expression

On the specified schedule (every day at 12:00pm 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`.

```
aws application-autoscaling put-scheduled-action --service-namespace dynamodb \
  --scalable-dimension dynamodb:table:WriteCapacityUnits \
  --resource-id table/my-table \
  --scheduled-action-name my-recurring-action \
  --schedule "cron(0 12 * * ? *)" \
  --scalable-target-action MinCapacity=10,MaxCapacity=50
```

Example Example: To scale on a recurring schedule using a rate expression

On the specified schedule (every hour), 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`.

```
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 "rate(1 hour)" \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

Describe Scheduled Actions

You can describe all scheduled actions for the specified service namespace using the following **describe-scheduled-actions** command.

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

The following is example output.

```
{
    "ScheduledActions": [
        {
            "ScheduledActionARN": "<arn>",
            "ServiceNamespace": "ecs",
            "CreationTime": 1515026382.218,
            "ScalableDimension": "ecs:service:DesiredCount",
            "Schedule": "at(2018-01-31T17:00:00)",
            "ScalableTargetAction": {
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-one-time-action",
            "ResourceId": "service/default/web-app"
        }
    ]
}
```

Delete a Scheduled Action

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

The following command deletes the specified scheduled action for the specified scalable target.

```
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-8788-1cee4EXAMPLE \
  --scheduled-action-name my-spot-fleet-action
```

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.

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`.

To suspend a scaling activity

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

```
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 `--suspendedstate` option as follows.

```
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-target** command to determine which scaling activities are in a suspended state for a scalable target.

```
aws application-autoscaling describe-scalable-target --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.

```
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
}
```

Authentication and Access Control for Application Auto Scaling

Access to Application Auto Scaling requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to perform Application Auto Scaling actions, such as configuring scaling policies.

This topic provides details on how you can use AWS Identity and Access Management (IAM) to help secure your resources by controlling who can perform Application Auto Scaling actions.

By default, a brand new IAM user has no permissions to do anything. To grant permissions to call Application Auto Scaling actions, you attach an IAM policy to the IAM users or groups that require the permissions it grants.

Specifying Actions in a Policy

Application Auto Scaling provides a set of actions that you can specify in an IAM policy. For more information, see Actions in the *Application Auto Scaling API Reference*.

To specify a single policy, you can use the following prefix with the name of the action: `application-autoscaling:`. For example:

```
"Action": "application-autoscaling:DescribeScalingActivities"
```

Wildcards are supported. For example, you can use `application-autoscaling:*` to specify all Application Auto Scaling actions.

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

You can also use `Describe*` to specify all actions whose names start with `Describe`.

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

In addition to the permissions for calling Application Auto Scaling actions, users also require permissions to create a service-linked role.

When users call `RegisterScalableTarget`, Application Auto Scaling creates a service-linked role 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 other services on your behalf.

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

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

For more information, see Service-Linked Roles for Application Auto Scaling (p. 32).

Specifying the Resource

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 you to use in an IAM policy. To control access to Application Auto Scaling actions, always use an * (asterisk) as the resource when writing an IAM policy.

Specifying Conditions in a Policy

When you grant permissions, you can use IAM policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. To express conditions, use predefined condition keys.

For a list of condition keys supported by each AWS service, see Actions, Resources, and Condition Keys for AWS Services in the *IAM User Guide*. For a list of condition keys that can be used in multiple AWS services, see AWS Global Condition Context Keys in the *IAM User Guide*.

Application Auto Scaling does not provide additional condition keys.

Example Policies

To configure step scaling policies or target tracking policies for a scalable resource, users must have permissions to use the actions in the following example policy:

```
{
    "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",
              "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*"
        }
    ]
}
```

To configure scheduled scaling for a scalable resource, users must have permissions to use the actions in the following example policy:

```
{
    "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",
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*"
        }
    ]
}
```

Additional IAM Permissions

Users must have additional permissions for each type of resource for which they will configure scaling policies. You specify the following actions in the `Action` element of an IAM policy statement.

AppStream 2.0 fleets

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

DynamoDB tables and global secondary indexes

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

Amazon EC2 Spot Fleet requests

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

Amazon ECS services

- `ecs:DescribeServices`
- `ecs:UpdateServices`
- `cloudwatch:DeleteAlarms`
- `cloudwatch:DescribeAlarms`
- `cloudwatch:PutMetricAlarm`

Amazon EMR clusters

- `elasticmapreduce:ModifyInstanceGroups`
- `elasticmapreduce:ListInstanceGroups`
- `cloudwatch:DeleteAlarms`
- `cloudwatch:DescribeAlarms`
- `cloudwatch:PutMetricAlarm`

Aurora DB clusters

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

Amazon SageMaker endpoints

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

Custom Resources

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

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. For more information, see Using Service-Linked Roles in the *IAM User Guide*.

You can delete the roles only after first deleting their related resources. This protects your resources because you can't inadvertently remove permissions to access the resources.

Permissions Granted by the Service-Linked Roles

Application Auto Scaling uses the following service-linked roles to manage scaling on your behalf. There is one service-linked role per type of scalable resource. In each case, the service-linked role is a predefined role that includes all the necessary permissions. Each service-linked role trusts the specified service principal to assume it.

`AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`

Actions:

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

Service principal: `appstream.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`

Actions:

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

Service principal: `dynamodb.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`

Actions:

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

Service principal: `ec2.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_ECSService`

Actions:

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

- `cloudwatch:PutMetricAlarm`

Service principal: `ecs.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_RDSCluster`

Actions:

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

Service principal: `rds.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`

Actions:

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

Service principal: `sagemaker.application-autoscaling.amazonaws.com`

`AWSServiceRoleForApplicationAutoScaling_CustomResource`

Actions:

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

Service principal: `custom-resource.application-autoscaling.amazonaws.com`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Using Service-Linked Roles in the *IAM User Guide*.

Create Service-Linked Roles (Automatic)

Under most circumstances, 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.

Important
The IAM user calling the `RegisterScalableTarget` action must have the appropriate IAM permissions to create the service-linked role. Otherwise, the automatic creation fails. For more information, see Service-Linked Role Permissions in the *IAM User Guide* or the information about required user permissions (p. 29) in this guide.

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*.

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 type of scalable resource, we recommend that you delete the corresponding service-linked role. You can delete a service-linked role only after first deleting the related scalable resources. For more information, see the documentation for the scalable resource. For example, to delete an ECS service, see Deleting a Service in the *Amazon Elastic Container Service Developer Guide*.

To delete service-linked roles, you can use the IAM console, the IAM CLI, or the IAM API. 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 Regions where the service is available. For more information, see AWS Regions and Endpoints.

Application Auto Scaling Limits

Your AWS account has the following limits related to Application Auto Scaling. To request a limit increase, use the Application Auto Scaling Limits form.

Default Limits Per Region Per Account

Item	Default Limit	Notes
Maximum number of scalable targets per resource type	Amazon DynamoDB: 3000 All other resource types: 500	Make sure that you specify the type of resource with your request for a limit increase, for example, Amazon ECS or DynamoDB.
Maximum number of scaling policies per scalable target	50	This 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	

For information about the service limits for other AWS services, see AWS Service Limits in the *Amazon Web Services General Reference*.

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
Suspend and resume scaling (p. 37)	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. 37)	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. 37)	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. 37)	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 Amazon SageMaker endpoint variants (p. 37)	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 with Aurora Replicas in the *Amazon RDS User Guide*.	November 17, 2017

Change	Description	Date
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 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