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:

- 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.
- Amazon Comprehend document classification endpoints
- Lambda function provisioned concurrency
- Amazon Keyspaces (for Apache Cassandra) tables

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

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.

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. For more information, see Getting Started with Application Auto Scaling (p. 12).

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 Application Auto Scaling 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 Application Auto Scaling 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 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. 39).

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. 25).
- 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. 13).

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
```

Getting Started 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. Automatic scaling is an optional feature of the service that is disabled by default in almost all cases.

With Application Auto Scaling, there are two ways to enable automatic scaling: using the service console of the resources that you want to scale, and accessing the Application Auto Scaling API programmatically or through the AWS Command Line Interface (AWS CLI). You can use either method, or both. In most cases, you can achieve the same result using either method.

If you are just getting started with Application Auto Scaling or are exploring the AWS Management Console, we recommend that you refer to the following documentation for sample configurations and to learn more about how the service integrates with Application Auto Scaling. If a service doesn't include documentation for configuring scaling in the console, you can use the AWS CLI or API to configure scaling.

- 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*
- Autoscaling with Endpoints in the *Amazon Comprehend Developer Guide*
- Managing Concurrency for a Lambda Function in the *AWS Lambda Developer Guide*
- Managing Amazon Keyspaces Throughput Capacity with Application Auto Scaling in the *Amazon Keyspaces (for Apache Cassandra) Developer Guide*

This user guide shows how to work with the Application Auto Scaling API using the AWS CLI. 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 CLI to access Application Auto Scaling as a first-time user. You learn the basic steps to configure scaling so your DynamoDB table scales at scheduled times. We chose to use a DynamoDB table as the basis for the tutorial because DynamoDB is available in most Regions, but you can use any supported service that you are comfortable with.

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

- Target tracking scaling policies are not supported for Amazon EMR.
- You cannot use the `RegisterScalableTarget` or `PutScalingPolicy` API operations to update an AWS Auto Scaling scaling plan. 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 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. 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 20 percent below the target value. 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.

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 with Application Auto Scaling (p. 12) 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.

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. 13). 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.

Limits

- Step scaling policies are not supported for DynamoDB, Amazon Comprehend, Lambda, or Amazon Keyspaces (for Apache Cassandra).

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

Lower Bound	Upper Bound	Adjustment
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. 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.

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 with Application Auto Scaling (p. 12) 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 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`:

```
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 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`.

```
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.

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 the scheduled action. It does not keep track of old values and return to the older values after the end time.

- The name of the scheduled action must be unique among all other scheduled actions on the specified scalable target.

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

```
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.

```
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 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. 33).

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 services when these events happen. For more information, see Application Auto Scaling Events and EventBridge (p. 35).

You can also write rules that trigger on API calls made by Application Auto Scaling. For more information, see Creating an EventBridge Rule That Triggers on an AWS API Call Using AWS CloudTrail in the *Amazon EventBridge User Guide*.

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 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 Personal Health Dashboard Notifications for Application Auto Scaling (p. 36).

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
CPUUtilization	AWS/ECS	ClusterName, ServiceName	ECS
MemoryUtilization	AWS/ECS	ClusterName, ServiceName	ECS
CPUUtilization	AWS/EC2Spot	FleetRequestId	Spot Fleet
NetworkIn	AWS/EC2Spot	FleetRequestId	Spot Fleet
NetworkOut	AWS/EC2Spot	FleetRequestId	Spot Fleet

Metric Name	Namespace	Dimensions	Applies To
RequestCountPerTarget	AWS/ ApplicationELB	TargetGroup	ECS, Spot Fleet
YARNMemoryAvailablePercentage	AWS/ ElasticMapReduce	ClusterId	EMR
AvailableCapacity	AWS/ AppStream	Fleet	AppStream
CapacityUtilization	AWS/ AppStream	Fleet	AppStream
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/RDS	DBClusterIdentifier, Role (READER)	Aurora
DatabaseConnections	AWS/RDS	DBClusterIdentifier, Role (READER)	Aurora
InvocationsPerInstance	AWS/ SageMaker	EndpointName, VariantName	SageMaker
InferenceUtilization	AWS/ Comprehend	EndpointArn	Comprehend
ProvisionedConcurrencyUtilization	AWS/ Lambda	FunctionName, Resource	Lambda
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

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 setup 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*.

Application Auto Scaling Events and 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. For example, you could be notified when a resource has scaled to its maximum capacity. 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 AWS SMS queue

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

The following is the format of Application Auto Scaling events with monitoring support in Amazon EventBridge.

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
    ]
  }
}
```

Personal Health Dashboard Notifications for Application Auto Scaling

To help you manage failed scaling events, your 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 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

Application Auto Scaling conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all of the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

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), so that 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 TLS to communicate with AWS resources.
- 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.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields or metadata, such as names and tags. Any data that you enter into metadata might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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 AWS 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 AWS 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 actions.

Topics

- Example Policies for Working in the AWS Management Console (p. 51)

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 (p. 40)
- Application Auto Scaling Resource-Based Policies (p. 41)
- Access Control Lists (ACLs) (p. 41)
- Authorization Based on Application Auto Scaling Tags (p. 41)
- Application Auto Scaling IAM Roles (p. 41)

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

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used 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 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 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 Example Policies for Working with the AWS CLI or an AWS SDK (p. 47).

Application Auto Scaling Resource-Based Policies

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. 42).

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 in Amazon EMR 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.

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*.

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 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 of 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_CassandraTable`

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_CustomResource

Actions:

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

Service principal: `custom-resource.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_LambdaConcurrency`

Actions:

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

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

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.

> **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 Permissions Required to Create a Service-Linked Role (p. 50) in this guide.

You automatically create a service-linked role using an AWS CloudFormation template by specifying the `RoleARN` property. To specify the service-linked role in the `RoleARN` property, use its full Amazon Resource Name (ARN). The ARN has the following general syntax:

```
arn:aws:[service]:[region]:[account]:resourceType/resourcePath
```

For more information, see Service-Linked Role ARN Reference (p. 46).

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 a value for "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.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 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

AWS CloudFormation requires you to specify an IAM role in your stack templates when registering scalable targets. The following service-linked role ARNs are defined by this service and can be used in the `RoleARN` property of an AWS::ApplicationAutoScaling::ScalableTarget in your stack template.

Resource Types	ARN
Amazon Keyspaces tables	`arn:aws:iam::012345678910:role/aws-service-role/cassandra.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable`
AppStream 2.0 fleets	`arn:aws:iam::012345678910:role/aws-service-role/appstream.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_AppStreamFleet`
Aurora DB clusters	`arn:aws:iam::012345678910:role/aws-service-role/rds.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_RDSCluster`
Comprehend document classification endpoints	`arn:aws:iam::012345678910:role/aws-service-role/comprehend.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ComprehendEndpoint`
Custom resources	`arn:aws:iam::012345678910:role/aws-service-role/custom-resource.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CustomResource`
DynamoDB tables and global secondary indexes	`arn:aws:iam::012345678910:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable`
EC2 Spot Fleet requests	`arn:aws:iam::012345678910:role/aws-service-role/ec2.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest`
ECS services	`arn:aws:iam::012345678910:role/aws-service-role/ecs.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_ECSService`

Resource Types	ARN
SageMaker endpoint variants	`arn:aws:iam::012345678910:role/aws-service-role/sagemaker.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint`
Lambda function provisioned concurrency	`arn:aws:iam::012345678910:role/aws-service-role/lambda.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_LambdaConcurrency`

Example Policies for Working with the AWS CLI or an AWS SDK

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 actions, such as configuring scaling policies. The administrator must then attach those policies to the IAM users or roles that require those permissions.

To learn how to create an IAM policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

The following policies grant the permissions for common use cases. You can attach these policies to your IAM users, based on the access that they need. Each policy grants access to all or some of the API actions for Application Auto Scaling.

The following shows an example of a permissions policy that allows a user to perform all API actions for Application Auto Scaling.

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

The following shows an example of a permissions policy that allows a user to configure scaling policies.

```
{
    "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 shows an example of a permissions policy that allows a user to configure scheduled scaling.

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

Additional Required 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.

ECS services

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

EC2 Spot Fleet requests

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

Amazon EMR clusters

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

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`

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`

Amazon Comprehend document classification endpoints

- `comprehend:UpdateEndpoint`
- `comprehend:DescribeEndpoint`
- `cloudwatch:DeleteAlarms`

- `cloudwatch:DescribeAlarms`
- `cloudwatch:PutMetricAlarm`

Lambda functions

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

Amazon Keyspaces tables

- `cassandra:Select`
- `cassandra:Alter`
- `cloudwatch:DeleteAlarms`
- `cloudwatch:DescribeAlarms`
- `cloudwatch:PutMetricAlarm`

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-specific 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"
```

The following shows an example of a permissions policy that allows a user to create an Application Auto Scaling service-linked role for Spot Fleet.

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

For more information, see Service-Linked Roles for Application Auto Scaling (p. 42).

Example Policies 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 actions in the AWS Management Console. You can use the example policies in the previous section; however, they are designed for requests that are made with the AWS CLI or an AWS 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*.

Keep in mind that Application Auto Scaling requires permission to create a service-linked role the first time that you configure scaling for a given service. For example, after this role has been created for the first scaling configuration you add to a Spot Fleet, you do not need to add this permission to other users of the same AWS account who use Spot Fleet.

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

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.

For more information, see Creating an Interface Endpoint in the *Amazon VPC User Guide*.

> **Important**
> Make sure that you're using the latest version of the CLI or SDK before creating an interface VPC endpoint. For more information, see the section called “Endpoint Migration” (p. 54).

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 public 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 create 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:PutScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

For more information, see Using VPC Endpoint Policies in the *Amazon VPC User Guide*.

Endpoint Migration

On November 22, 2019, Application Auto Scaling 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`.

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 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*.

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
Add support for Amazon Keyspaces (for Apache Cassandra) tables (p. 57)	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. 57)	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. 57)	Various improvements and corrections.	January 15, 2020
Add notification functionality (p. 57)	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. 57)	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. 57)	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. 57)	Use target tracking scaling policies to scale the size of an AppStream 2.0 fleet.	November 25, 2019

Support for Amazon VPC endpoints (p. 57)	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. 57)	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. 57)	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. 57)	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. 57)	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. 57)	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
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

Change	Description	Date
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