Application Auto Scaling

User Guide

Application Auto Scaling: User Guide

Table of Contents

What Is Application Auto Scaling?

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. More information is available in our GitHub repository.

You have several options for scaling with AWS. For information on 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. 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

AWS provides a web-based user interface, the AWS Management Console. If you've signed up for an AWS account, you can access Application Auto Scaling by signing into the AWS Management Console and opening the service console for your scalable resource.

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 about the commands for Application Auto Scaling, 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 about the cmdlets for Application Auto Scaling, see the AWS Tools for PowerShell Cmdlet Reference.

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 about the API actions for Application Auto Scaling, 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.

For information about your credentials for accessing AWS, see AWS Security Credentials in the *Amazon Web Services General Reference*.

Getting Started

To learn more about the AWS services supported by Application Auto Scaling, see the following documentation:

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

To see the regional availability for specific AWS services, see the AWS Region Table.

For information about regions and endpoints for calls to the Application Auto Scaling API or CLI, visit AWS Regions and Endpoints in the *AWS General Reference*.

Target Tracking Scaling Policies for Application Auto Scaling

With target tracking scaling policies, you select a predefined metric or configure a customized 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 and minimizes changes to the capacity of the scalable target.

When specifying a customized metric, be aware that not all metrics work for target tracking. 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.

You can have multiple target tracking scaling policies for a scalable target, provided that each of them uses a different metric. Application Auto Scaling scales based on the policy that provides the largest capacity for both scale in and scale out. This provides greater flexibility to cover multiple scenarios and ensures that there is always enough capacity to process your application workloads.

You can also optionally 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, a different scaling policy type).

Limits

- You cannot create target tracking scaling policies for Amazon EMR clusters or AppStream 2.0 fleets.
- You can create 50 scaling policies per scalable target. This includes both step scaling policies and target tracking policies.

Considerations

Keep the following considerations in mind:

- A target tracking scaling policy assumes that it should perform scale out when the specified metric is above the target value. You cannot use a target tracking scaling policy to scale out when the specified metric is below the target value.
- A target tracking scaling policy does not perform scaling when the specified metric has insufficient data. It does not perform scale in because it does not interpret insufficient data as low utilization. To scale in when a metric has insufficient data, create a step scaling policy and have an alarm invoke the scaling policy when it changes to the `INSUFFICIENT_DATA` state. For information about alarm states, see Alarm States in the *Amazon CloudWatch User Guide*.
- 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 small capacity, the actual metric data points might seem far from the target value. For a scalable target with larger capacity, adding or removing capacity causes less of a gap between the target value and the actual metric data points.

- We recommend that you scale based on metrics with a 1-minute frequency because that ensures a faster response to utilization changes. Scaling on metrics with a 5-minute frequency can result in slower response time and scaling on stale metric data.
- To ensure application availability, Application Auto Scaling scales out proportionally to the metric as fast as it can, but scales in more gradually.
- Do not edit or delete the CloudWatch alarms that Application Auto Scaling manages for a target tracking scaling policy. Application Auto Scaling deletes the alarms automatically when you delete the Auto Scaling policy.

Cooldown Period

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

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

Create a Target Tracking Scaling Policy

You can create scaling policies that tell Application Auto Scaling what to do when the specified conditions change. Before you can create a scaling policy, you must register a scalable target.

Use the following register-scalable-target command to register a scalable target.

```
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
```

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

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

Use the following put-scaling-policy command, along with the `config.json` file you created, to create a scaling policy named `cpu40` that keeps the average CPU utilization of the specified scalable target at 40 percent:

```
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 --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",
    "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. Note that 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": "<arn>",
        "TargetTrackingScalingPolicyConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "EC2SpotFleetRequestAverageCPUUtilization"
            },
            "TargetValue": 40.0
        },
        "PolicyName": "cpu40",
        "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": "<arn>"
            },
            {
                "AlarmName": "TargetTracking-spot-fleet-request/sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE-AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
                "AlarmARN": "<arn>"
            }
        ],
        "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
```

Step Scaling Policies for Application Auto Scaling

Step scaling policies increase or decrease the current capacity of a scalable target based on a set of scaling adjustments, known as *step adjustments*, that vary based on the size of the alarm breach.

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

Limits

- You cannot create step scaling policies for DynamoDB tables and global secondary indexes.
- You can create 50 scaling policies per scalable target. This includes both step scaling policies and target tracking policies.
- You can create 20 step adjustments per scaling policy.

Scaling Adjustment Types

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

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

- **ChangeInCapacity**—Increase or decrease the current capacity of the scalable target by the specified value. A positive value increases the capacity and a negative value decreases the capacity.

  Example: If the current capacity of a Spot Fleet is 3 and the adjustment is 5, then when this policy is performed, Application Auto Scaling adds 5 to the capacity of the Spot Fleet for a total of 8.
- **ExactCapacity**—Change the current capacity of the scalable target to the specified value. Specify a positive value with this adjustment type.

  Example: If the current capacity of a Spot Fleet is 3 and the adjustment is 5, then when this policy is performed, Application Auto Scaling changes the capacity to 5.
- **PercentChangeInCapacity**—Increase or decrease the current capacity of the scalable target by the specified percentage. A positive value increases the capacity and a negative value decreases the capacity. If the resulting value is not an integer, Application Auto Scaling rounds it as follows:
  - Values greater than 1 are rounded down. For example, `12.7` is rounded to `12`.
  - Values between 0 and 1 are rounded to 1. For example, `.67` is rounded to `1`.
  - Values between 0 and -1 are rounded to -1. For example, `-.58` is rounded to `-1`.
  - Values less than -1 are rounded up. For example, `-6.67` is rounded to `-6`.

  Example: If the current capacity is 10 and the adjustment is 10 percent, then when this policy is performed, Application Auto Scaling adds 1 to the capacity for a total of 11.

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

Step Adjustments

When you create a step scaling policy, you add one or more step adjustments that enable you to scale based on the size of the alarm breach. Each step adjustment specifies a lower bound for the metric value, an upper bound for the metric value, and the amount by which to scale, based on the scaling adjustment type.

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

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

Application Auto Scaling applies the aggregation type to the metric data points from all scalable targets. It compares the aggregated metric value against the upper and lower bounds defined by the step adjustments to determine which step adjustment to perform.

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

Scale out policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
0	10	0	50 <= *value* < 60
10	20	10	60 <= *value* < 70
20	null	30	70 <= *value* < +infinity
Scale in policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
-10	0	0	40 < *value* <= 50
-20	-10	-10	30 < *value* <= 40
null	-20	-30	-infinity < *value* <= 30

Your scalable target has both a current capacity and a desired capacity of 10. The current and desired capacity is maintained while the aggregated metric value is greater than 40 and less than 60.

If the metric value gets to 60, Application Auto Scaling increases the desired capacity of the group by 1, to 11, 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, based on the third step adjustment of the scale-out policy (add 30 percent of 11, 3.3, rounded down to 3).

If the metric value gets to 40, Application Auto Scaling decreases the target capacity by 1, to 13, based on the second step adjustment of the scale-in policy (remove 10 percent of 14, 1.4, rounded down to 1). If the metric value falls to 30 even after this decrease in capacity, Application Auto Scaling decreases the target capacity by 3, to 10, based on the third step adjustment of the scale-in policy (remove 30 percent of 13, 3.9, rounded down to 3).

Cooldown Period

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

For scale out policies, while the cooldown period is in effect, the capacity that has been added by the previous scale out event that initiated the cooldown is calculated as part of the desired capacity for the next scale out event. The intention is to continuously (but not excessively) scale out. For example, when an alarm triggers a step scaling policy to increase the capacity by 2, the scaling activity completes successfully and a cooldown period starts. During the cooldown period, if the alarm triggers the same policy again but at a more aggressive step adjustment, say by 3, the increase of 2 from the previous scale out event is considered part of the current capacity. Therefore, only 1 is added to the capacity.

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

Create a Step Scaling Policy

You can create scaling policies that tell Application Auto Scaling what to do when the specified conditions change. Before you can create a scaling policy, you must register the scalable target.

Use the following register-scalable-target command to register a scalable target.

```
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
```

The following is an example step configuration with three step adjustments. Save this configuration in a file named `config.json`.

```
{
  "AdjustmentType": "ChangeInCapacity",
  "Cooldown": 60,
  "StepAdjustments": [
    {
      "MetricIntervalLowerBound": 0,
      "MetricIntervalUpperBound": 10,
      "ScalingAdjustment": 0
    },
    {
      "MetricIntervalLowerBound": 10,
      "MetricIntervalUpperBound": 20,
      "ScalingAdjustment": 1
    },
    {
      "MetricIntervalLowerBound": 20,
      "ScalingAdjustment": 3
    }
  ]
}
```

Use the following put-scaling-policy command, along with the `config.json` file you created, to create a scaling policy named `my-step-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 my-step-scaling-policy --policy-type StepScaling \
--step-scaling-policy-configuration file://config.json
```

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 ec2
```

You can filter the results to just the step scaling policies using the `--query` parameter. Note that 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==`StepScaling`]'
```

The following is example output.

```
[
    {
        "PolicyARN": "<arn>",
        "StepScalingPolicyConfiguration": {
            "Cooldown": 60,
            "StepAdjustments": [
                {
                    "MetricIntervalLowerBound": 0.0,
                    "ScalingAdjustment": 0,
                    "MetricIntervalUpperBound": 10.0
                },
                {
                    "MetricIntervalLowerBound": 10.0,
                    "ScalingAdjustment": 10,
                    "MetricIntervalUpperBound": 20.0
                },
                {
                    "MetricIntervalLowerBound": 20.0,
                    "ScalingAdjustment": 30
                }
            ],
            "AdjustmentType": "ChangeInCapacity"
        },
        "PolicyType": "StepScaling",
        "ResourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
        "ServiceNamespace": "ec2",
        "Alarms": [],
        "PolicyName": "my-step-scaling-policy",
        "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
        "CreationTime": 1515024099.901
    }
]
```

Delete a Step Scaling Policy

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

The following command deletes the specified step scaling policy for the specified Spot fleet request.

```
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 my-step-scaling-policy
```

Scheduled Scaling for Application Auto Scaling

Scaling based on a schedule enables you to scale your application in response to predictable load changes. You can plan scaling activities based on the predictable traffic patterns of your web application. 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, you 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, the minimum capacity, and the 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.

Create a Scheduled Action

You can create a scheduled action to scale one time only or 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 a minimum and a maximum capacity.

Example Example: To scale one time only

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

```
aws application-autoscaling put-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-one-time-action \
--schedule at(2018-01-31T17:00:00) \
--scalable-target-action MaxCapacity=3
```

Example Example: To scale on a recurring schedule

On the specified schedule, if the current capacity is below the value specified for `MinCapacity`, Application Auto Scaling scales out to the value specified by `MinCapacity`.

```
aws application-autoscaling put-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-recurring-action \
--schedule cron(expression) \
--scalable-target-action MinCapacity=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 ec2
```

The following is example output.

```
{
    "ScheduledActions": [
        {
            "ScheduledActionARN": "<arn>",
            "ServiceNamespace": "ec2",
            "CreationTime": 1515026382.218,
            "ScalableDimension": "ec2:spot-fleet-request:TargetCapacity",
            "Schedule": "at(2018-01-31T17:00:00)",
            "ScalableTargetAction": {
                "MaxCapacity": 3
            },
            "ScheduledActionName": "my-one-time-action",
            "ResourceId": "spot-fleet-request/sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
        }
    ]
}
```

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 Spot fleet request.

```
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
```

Authentication and Access Control for Application Auto Scaling

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

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

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

Specifying Actions in a Policy

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

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

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

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

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

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

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

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

When users call `RegisterScalableTarget`, Application Auto Scaling creates a service-linked role in your account, if the role does not exist already. The service-linked role grants permissions to Application Auto Scaling, so that it can call other services on your behalf.

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

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

For more information, see Service-Linked Roles for Application Auto Scaling (p. 17).

Specifying the Resource

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

Specifying Conditions in a Policy

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

For a list of context keys supported by each AWS service and a list of AWS-wide policy keys, see Actions, Resources, and Condition Keys for AWS Services and AWS Global Condition Context Keys in the *IAM User Guide*.

Application Auto Scaling does not provide additional condition keys.

Example Policies

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

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

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "application-autoscaling:RegisterScalableTarget",
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DeregisterScalableTarget",
                "application-autoscaling:PutScheduledAction",
                "application-autoscaling:DescribeScheduledActions",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DeleteScheduledAction",
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*"
        }
    ]
}
```

Additional IAM Permissions

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

AppStream 2.0 fleets

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

DynamoDB tables and global secondary indexes

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

Amazon EC2 Spot Fleet requests

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

Amazon ECS services

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

Amazon EMR clusters

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

Aurora DB clusters

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

Amazon SageMaker endpoints

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

Custom Resources

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

Service-Linked Roles for Application Auto Scaling

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

Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. For more information, see Using Service-Linked Roles in the *IAM User Guide*.

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

Permissions Granted by the Service-Linked Roles

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

AWSServiceRoleForApplicationAutoScaling_AppStreamFleet

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_DynamoDBTable

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_EC2SpotFleetRequest

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_ECSService

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_RDSCluster

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_SageMakerEndpoint

Actions:

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

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

AWSServiceRoleForApplicationAutoScaling_CustomResource

Actions:

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

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

Create Service-Linked Roles (Automatic)

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

> **Important**
> The IAM user calling the `RegisterScalableTarget` action must have the appropriate IAM permissions to create the service-linked role. Otherwise, the automatic creation fails. For more information, see Service-Linked Role Permissions in the *IAM User Guide*.

Create Service-Linked Roles (Manual)

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

Edit the Service-Linked Roles

Because various entities might reference a service-linked role, you cannot change the name of the role.

However, you can edit the description of a service-linked role created for Application Auto Scaling using IAM. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Delete the Service-Linked Roles

If you no longer use Application Auto Scaling with a type of scalable resource, we recommend that you delete the corresponding service-linked role. You can delete a service-linked role only after first deleting the related scalable resources. For more information, see the documentation for the scalable resource. For example, to delete an ECS service, see Deleting a Service in the *Amazon Elastic Container Service Developer Guide*.

To delete service-linked roles, you can use the IAM console, the IAM CLI, or the IAM API . For more information, see Deleting a Service-Linked Role in the *IAM User Guide*.

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

Application Auto Scaling Limits

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

- Scalable targets: 500
- Scaling policies per scalable target: 50
- Scheduled actions per scalable target: 200
- Step adjustments per scaling policy: 20

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 custom resources (p. 22)	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. 22)	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
Add support for target tracking scaling policies	Use target tracking scaling policies to set up dynamic scaling for your application in just a few simple steps. For more information, see Target Tracking Scaling Policies for Application Auto Scaling.	July 12, 2017
Add support for provisioned read and write capacity for DynamoDB tables and global secondary indexes	Use Application Auto Scaling to scale provisioned throughput capacity. For more information, see Managing Throughput Capacity with DynamoDB Auto Scaling in the *Amazon DynamoDB Developer Guide*.	June 14, 2017

Change	Description	Date
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 Developer 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