
Amazon EC2 Auto Scaling

User Guide



Amazon EC2 Auto Scaling: User Guide

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon EC2 Auto Scaling?	1
Auto Scaling components	1
Pricing for Amazon EC2 Auto Scaling	2
Get started	2
Related services	2
Work with Auto Scaling groups	2
Auto scaling benefits	3
Example: Cover variable demand	3
Example: Web app architecture	4
Example: Distribute instances across Availability Zones	5
Instance lifecycle	7
Scale out	8
Instances in service	8
Scale in	9
Attach an instance	9
Detach an instance	10
Lifecycle hooks	10
Enter and exit standby	10
Quotas	10
Set up	13
Prepare to use Amazon EC2	13
Install the AWS CLI	13
Get started	14
Walkthrough summary	14
Prepare for the walkthrough	15
Step 1: Create a launch template	15
Step 2: Create a single-instance Auto Scaling group	17
Step 3: Verify your Auto Scaling group	17
Step 4: Terminate an instance in your Auto Scaling group	18
Step 5: Next steps	18
Step 6: Clean up	19
Launch templates	21
Permissions	22
Create a launch template for an Auto Scaling group	22
Create your launch template (console)	23
Create a launch template from an existing instance (console)	29
Additional information	29
Limitations	30
Copy launch configurations to launch templates	30
Replace a launch configuration with a launch template	31
Request Spot Instances	32
AWS CLI examples for working with launch templates	33
Example usage	33
Create a basic launch template	34
Specify tags that tag instances at launch	35
Specify an IAM role to pass to instances	35
Assign public IP addresses	35
Specify a user data script that configures instances at launch	35
Specify a block device mapping	36
Specify Dedicated Hosts to bring software licenses from external vendors	36
Specify an existing network interface	36
Create multiple network interfaces	36
Manage your launch templates	37
Update an Auto Scaling group to use a launch template	39

Launch configurations	40
Create a launch configuration	40
Create a launch configuration (console)	41
Create a launch configuration (AWS CLI)	42
Configure IMDS	42
Create a launch configuration using an EC2 instance	44
Create a launch configuration using an EC2 instance	44
Create a launch configuration from an instance and override the block devices (AWS CLI)	45
Create a launch configuration and override the instance type (AWS CLI)	46
Change a launch configuration	47
Configure instance tenancy	48
Auto Scaling groups	51
Use multiple instance types and purchase options	52
Allocation strategies	53
Control the proportion of On-Demand Instances	55
Best practices for Spot Instances	56
Prerequisites	57
Create an Auto Scaling group with Spot and On-Demand Instances (console)	57
Configure Spot allocation strategies (AWS CLI)	59
Verify whether your Auto Scaling group is configured correctly and that the group has launched instances (AWS CLI)	64
Configure overrides	64
Create a group using attribute-based instance type selection	75
Considerations	75
Prerequisites	76
Use attribute-based instance type selection	77
Limitations	80
Create a group using a launch template	81
Create a group using a launch configuration	83
Create a group using an EC2 instance	85
Prerequisites	86
Create an Auto Scaling group from an EC2 instance (console)	86
Create an Auto Scaling group from an EC2 instance (AWS CLI)	86
Create a group using the EC2 launch wizard	89
Create a launch template	90
Create an Auto Scaling group	90
Next steps	91
Tag groups and instances	91
Tag naming and usage restrictions	92
EC2 instance tagging lifecycle	92
Tag your Auto Scaling groups	92
Delete tags	94
Tags for security	95
Control access to tags	96
Use tags to filter Auto Scaling groups	96
Replace instances	99
Replace instances based on an instance refresh	99
Check instance refresh status	106
AWS CLI examples that enable skip matching	107
Add checkpoints to an instance refresh	111
Replace instances based on maximum instance lifetime	114
Delete your Auto Scaling infrastructure	117
Delete your Auto Scaling group	117
(Optional) Delete the launch configuration	118
(Optional) Delete the launch template	118
(Optional) Delete the load balancer and target groups	119
(Optional) Delete CloudWatch alarms	119

Scale your group	121
Scaling options	121
Set capacity limits	122
Maintain a fixed number of instances	123
Manual scaling	124
Change the size of your Auto Scaling group (console)	124
Change the size of your Auto Scaling group (AWS CLI)	125
Attach EC2 instances to your Auto Scaling group	126
Detach EC2 instances from your Auto Scaling group	130
Dynamic scaling	133
How dynamic scaling policies work	133
Multiple dynamic scaling policies	134
Target tracking scaling policies	135
Step and simple scaling policies	141
Set default warm-up or cooldown values	150
Scaling based on Amazon SQS	157
Verify a scaling activity	161
Disable a scaling policy	163
Delete a scaling policy	164
AWS CLI examples for scaling policies	166
Predictive scaling	168
How predictive scaling works	168
Best practices	169
Create a predictive scaling policy (console)	169
Create a predictive scaling policy (AWS CLI)	172
Limitations	174
Supported Regions	174
Explore your data and forecast	175
Override the forecast	179
Use custom metrics	183
Scheduled scaling	190
Considerations	190
Recurring schedules	191
Create and manage scheduled actions (console)	191
Create and manage scheduled actions (AWS CLI)	193
Limitations	195
Lifecycle hooks	195
Considerations and limitations for lifecycle hooks	196
Lifecycle hook availability	197
Examples	197
How lifecycle hooks work	197
Prepare to add a lifecycle hook	199
Add lifecycle hooks	205
Complete a lifecycle action	207
Tutorial: Configure user data to retrieve the target lifecycle state through instance metadata	208
Tutorial: Configure a lifecycle hook that invokes a Lambda function	213
Warm pools	219
Core concepts	220
Prerequisites	221
Create a warm pool	222
Update a warm pool	222
Delete a warm pool	223
Limitations	223
Use lifecycle hooks	223
View health check status	226
AWS CLI examples for working with warm pools	228
Control instance termination	230

Termination policy scenarios	231
Work with termination policies	233
Create a custom termination policy with Lambda	236
Use instance scale-in protection	240
Temporarily remove instances	242
How the standby state works	243
Health status of an instance in a standby state	244
Temporarily remove an instance (console)	244
Temporarily remove an instance (AWS CLI)	244
Suspend-resume processes	247
Types of processes	247
Considerations	248
Suspend and resume processes (console)	250
Suspend and resume processes (AWS CLI)	250
Monitor	252
Check instance health	253
Health check types	253
Amazon EC2 health checks	253
Elastic Load Balancing health checks	254
Custom health detection tasks	255
Unhealthy instance replacement	256
How Amazon EC2 Auto Scaling minimizes downtime	256
Health check considerations	257
Additional information	257
Health check grace period	258
Monitor with AWS Health Dashboard	259
Monitor with CloudWatch	260
Available metrics and dimensions	261
Enable Auto Scaling group metrics (console)	264
Enable Auto Scaling group metrics (AWS CLI)	265
Configure monitoring for Auto Scaling instances	265
View monitoring graphs in the Amazon EC2 Auto Scaling console	267
Log API calls with AWS CloudTrail	269
Amazon EC2 Auto Scaling information in CloudTrail	269
Understand Amazon EC2 Auto Scaling log file entries	270
Monitor with Amazon SNS notifications	271
SNS notifications	271
Configure Amazon SNS notifications for Amazon EC2 Auto Scaling	272
Work with other services	275
Capacity Rebalancing	275
How it works	275
Considerations	276
Enable Capacity Rebalancing	277
Add a termination lifecycle hook	281
AWS CloudFormation	282
Amazon EC2 Auto Scaling and AWS CloudFormation templates	282
Learn more about AWS CloudFormation	283
AWS CloudShell	283
Compute Optimizer	283
Limitations	284
Findings	284
View recommendations	284
Considerations for evaluating the recommendations	285
Elastic Load Balancing	286
Elastic Load Balancing types	286
Prerequisites	287
Attach a load balancer	288

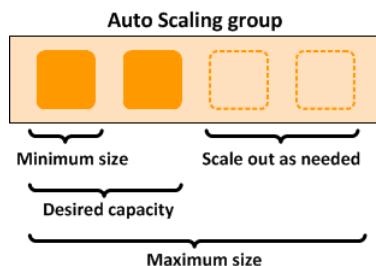
Understand the attachment status	290
Configure a load balancer from the Amazon EC2 Auto Scaling console	290
Add Elastic Load Balancing health checks	291
Add Availability Zones	292
AWS CLI examples for working with Elastic Load Balancing	294
Tutorial: Set up a scaled and load-balanced application	297
EventBridge	304
Amazon EC2 Auto Scaling event reference	304
Warm pool event types and patterns	309
Create EventBridge rules	314
Amazon VPC	317
EC2-Classic	318
Default VPC	318
Nondefault VPC	319
Considerations when choosing VPC subnets	319
IP addressing in a VPC	319
Network interfaces in a VPC	320
Instance placement tenancy	320
AWS Outposts	320
More resources for learning about VPCs	320
Security	322
Data protection	322
Use AWS KMS keys to encrypt Amazon EBS volumes	323
Identity and Access Management	323
Access control	324
How Amazon EC2 Auto Scaling works with IAM	324
Managed policies	330
Service-linked roles	332
Identity-based policy examples	337
Cross-service confused deputy prevention	346
Launch template support	347
IAM role for applications that run on Amazon EC2 instances	351
AWS KMS key policy for use with encrypted volumes	353
Compliance validation	357
PCI DSS compliance	358
Resilience	358
Infrastructure security	358
Use VPC endpoints for private connectivity	358
Create an interface VPC endpoint	359
Create a VPC endpoint policy	359
Troubleshoot	360
Retrieve an error message	360
Additional troubleshooting resources	361
Instance launch failure	362
The requested configuration is currently not supported.	363
The security group <name of the security group> does not exist. Launching EC2 instance failed.	363
The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed.	364
The requested Availability Zone is no longer supported. Please retry your request....	364
Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)...	364
Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735...	364
Invalid device name <device name> / Invalid device name upload. Launching EC2 instance failed.	365
Value (<name associated with the instance storage device>) for parameter virtualName is invalid...	365

EBS block device mappings not supported for instance-store AMIs.	365
Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.	366
Client.InternalError: Client error on launch.	366
We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed.	367
There is no Spot capacity available that matches your request. Launching EC2 instance failed.	367
<number of instances> instance(s) are already running. Launching EC2 instance failed.	367
AMI issues	368
The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.	368
AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.	368
Value (<ami ID>) for parameter virtualName is invalid.	368
The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching EC2 instance failed.	369
Load balancer issues	369
One or more target groups not found. Validating load balancer configuration failed.	370
Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed.	370
There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.	370
EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.	370
EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed.	371
Launch template issues	371
You must use a valid fully-formed launch template (invalid value)	371
You are not authorized to use launch template (insufficient IAM permissions)	372
Health checks	372
An instance was taken out of service in response to an EC2 instance status check failure	373
An instance was taken out of service in response to an EC2 scheduled reboot	373
An instance was taken out of service in response to an EC2 health check that indicated it had been terminated or stopped	374
An instance was taken out of service in response to an ELB system health check failure	374
Resources	376
Document history	377

What is Amazon EC2 Auto Scaling?

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups*. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



For more information about the benefits of Amazon EC2 Auto Scaling, see [Amazon EC2 Auto Scaling benefits \(p. 3\)](#).

Auto Scaling components

The following table describes the key components of Amazon EC2 Auto Scaling.

	Groups Your EC2 instances are organized into <i>groups</i> so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and, desired number of EC2 instances. For more information, see Auto Scaling groups (p. 51) .
	Configuration templates Your group uses a <i>launch template</i> , or a <i>launch configuration</i> (not recommended, offers fewer features), as a configuration template for its EC2 instances. You can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances. For more information, see Launch templates (p. 21) and Launch configurations (p. 40) .
	Scaling options Amazon EC2 Auto Scaling provides several ways for you to scale your Auto Scaling groups. For example, you can configure a group

to scale based on the occurrence of specified conditions (dynamic scaling) or on a schedule. For more information, see [Scaling options \(p. 121\)](#).

Pricing for Amazon EC2 Auto Scaling

There are no additional fees with Amazon EC2 Auto Scaling, so it's easy to try it out and see how it can benefit your AWS architecture. You only pay for the AWS resources (for example, EC2 instances, EBS volumes, and CloudWatch alarms) that you use.

Get started

To begin, complete the [Get started with Amazon EC2 Auto Scaling \(p. 14\)](#) tutorial to create an Auto Scaling group and see how it responds when an instance in that group terminates.

Related services

To automatically distribute incoming application traffic across multiple instances in your Auto Scaling group, use Elastic Load Balancing. For more information, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#).

To monitor your Auto Scaling groups and instance utilization data, use Amazon CloudWatch. For more information, see [Monitor CloudWatch metrics for your Auto Scaling groups and instances \(p. 260\)](#).

To configure auto scaling for scalable resources for Amazon Web Services beyond Amazon EC2, see the [Application Auto Scaling User Guide](#).

Work with Auto Scaling groups

You can create, access, and manage your Auto Scaling groups using any of the following interfaces:

- **AWS Management Console** – Provides a web interface that you can use to access your Auto Scaling groups. If you've signed up for an AWS account, you can access your Auto Scaling groups by signing into the AWS Management Console, using the search box on the navigation bar to search for **Auto Scaling groups**, and then choosing **Auto Scaling groups**.
- **AWS Command Line Interface (AWS CLI)** – Provides commands for a broad set of AWS services, and is supported on Windows, macOS, and Linux. To get started, see [AWS Command Line Interface User Guide](#). For more information, see `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](#).
- **AWS SDKs** – Provides language-specific API operations and takes care of many of the connection details, such as calculating signatures, handling request retries, and handling errors. For more information, see [AWS SDKs](#).
- **Query API** – Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access AWS services. However, it requires your application to handle low-level details such as generating the hash to sign the request, and handling errors. For more information, see the [Amazon EC2 Auto Scaling API Reference](#).
- **AWS CloudFormation** – Supports creating Auto Scaling groups using CloudFormation templates. For more information, see [Create Auto Scaling groups with AWS CloudFormation \(p. 282\)](#).

To connect programmatically to an AWS service, you use an endpoint. For information about endpoints for calls to Amazon EC2 Auto Scaling, see [Amazon EC2 Auto Scaling endpoints and quotas](#) in the [AWS General Reference](#).

Amazon EC2 Auto Scaling benefits

Adding Amazon EC2 Auto Scaling to your application architecture is one way to maximize the benefits of the AWS Cloud. When you use Amazon EC2 Auto Scaling, your applications gain the following benefits:

- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are needed and terminating them when they aren't.

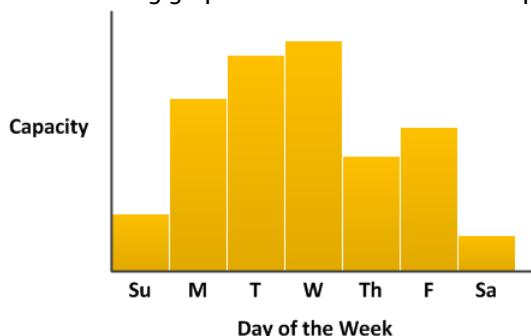
Contents

- [Example: Cover variable demand \(p. 3\)](#)
- [Example: Web app architecture \(p. 4\)](#)
- [Example: Distribute instances across Availability Zones \(p. 5\)](#)
 - [Instance distribution \(p. 6\)](#)
 - [Rebalancing activities \(p. 6\)](#)

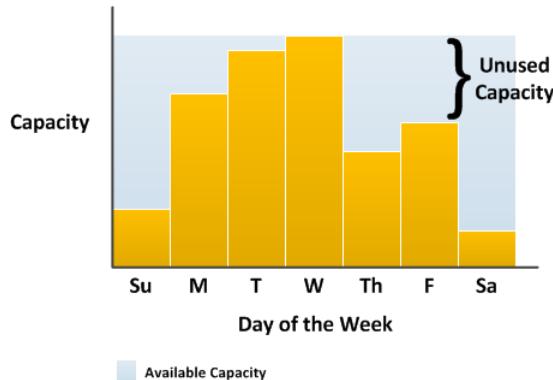
Example: Cover variable demand

To demonstrate some of the benefits of Amazon EC2 Auto Scaling, consider a basic web application running on AWS. This application allows employees to search for conference rooms that they might want to use for meetings. During the beginning and end of the week, usage of this application is minimal. During the middle of the week, more employees are scheduling meetings, so the demand on the application increases significantly.

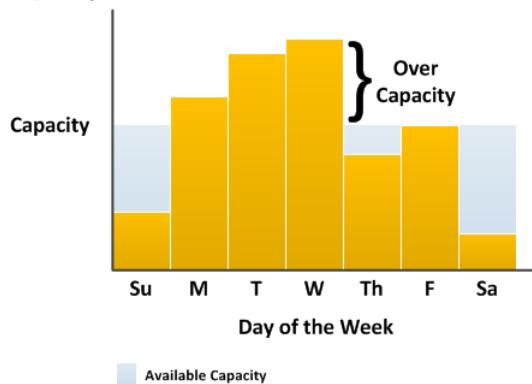
The following graph shows how much of the application's capacity is used over the course of a week.



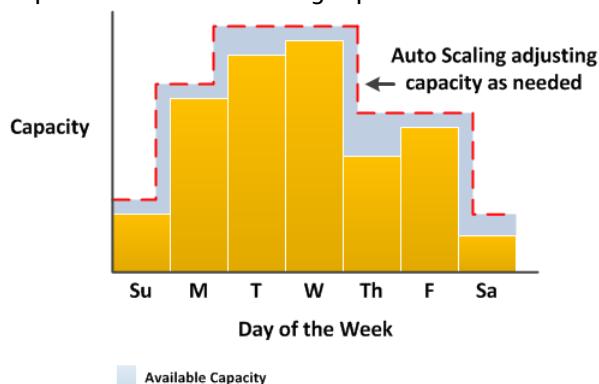
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity. The extra capacity remains unused and, in essence, raises the cost of keeping the application running.



The second option is to have enough capacity to handle the average demand on the application. This option is less expensive, because you aren't purchasing equipment that you'll only use occasionally. However, you risk creating a poor customer experience when the demand on the application exceeds its capacity.

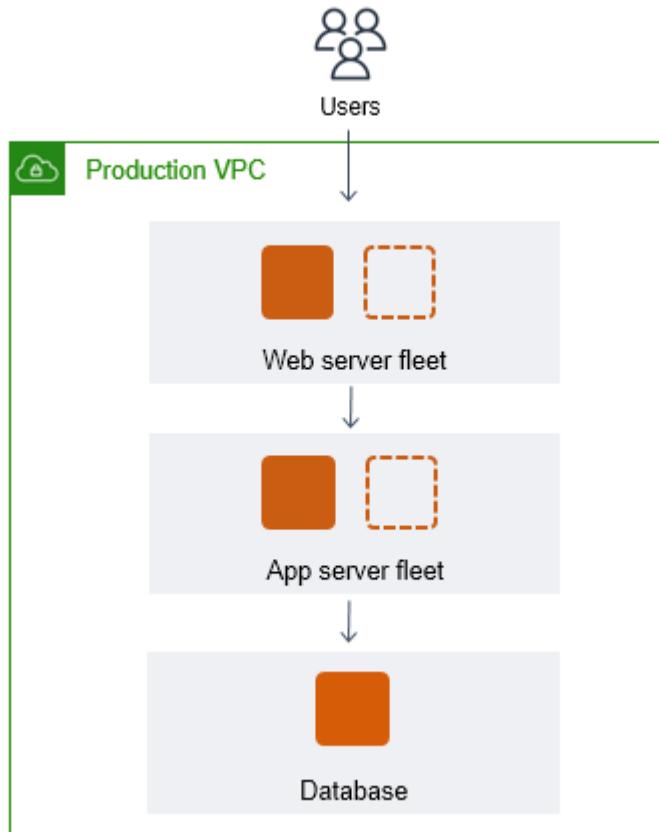


By adding Amazon EC2 Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. Because Amazon EC2 Auto Scaling uses EC2 instances, you only have to pay for the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while minimizing expenses.



Example: Web app architecture

In a common web app scenario, you run multiple copies of your app simultaneously to cover the volume of your customer traffic. These multiple copies of your application are hosted on identical EC2 instances (cloud servers), each handling customer requests.



Amazon EC2 Auto Scaling manages the launch and termination of these EC2 instances on your behalf. You define a set of criteria (such as an Amazon CloudWatch alarm) that determines when the Auto Scaling group launches or terminates EC2 instances. Adding Auto Scaling groups to your network architecture helps make your application more highly available and fault tolerant.

You can create as many Auto Scaling groups as you need. For example, you can create an Auto Scaling group for each tier.

To distribute traffic between the instances in your Auto Scaling groups, you can introduce a load balancer into your architecture. For more information, see [Elastic Load Balancing \(p. 286\)](#).

Example: Distribute instances across Availability Zones

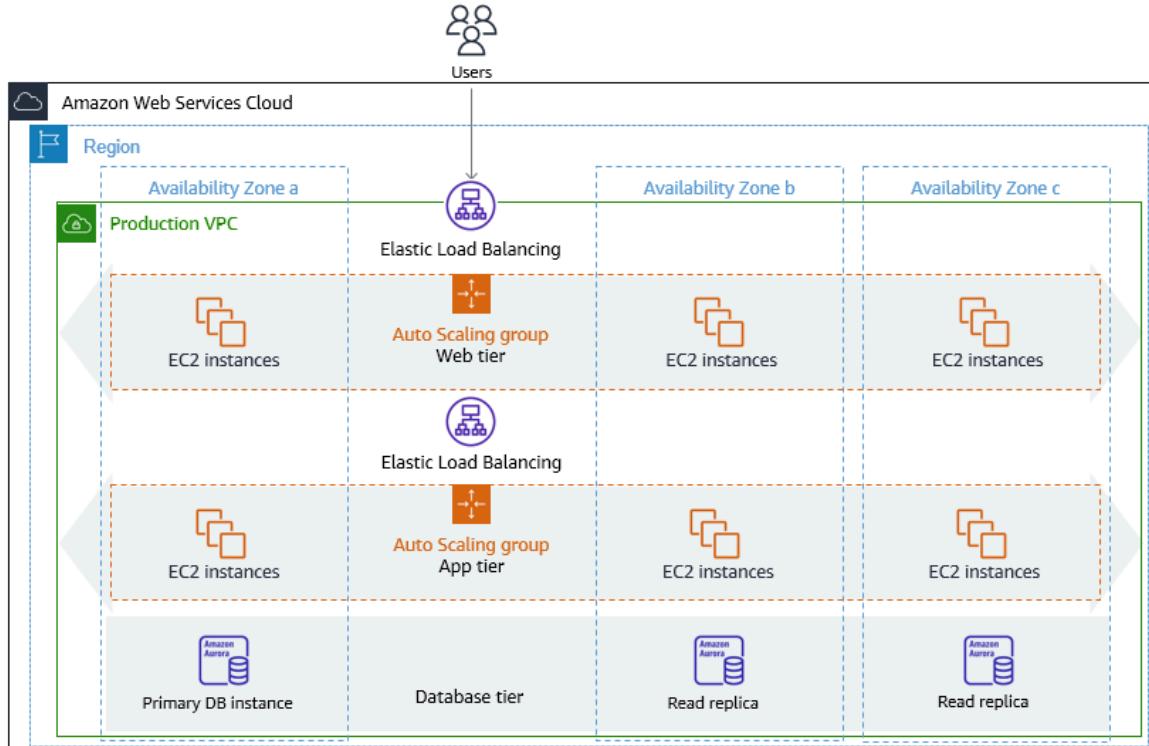
Availability Zones are isolated locations in a given AWS Region. Each Region has multiple Availability Zones designed to provide high availability for the Region. Availability Zones are independent, and therefore you increase application availability when you design your application to use multiple zones.

An Availability Zone is identified by the AWS Region code followed by a letter identifier (for example, us-east-1a). If you create your VPC and subnets rather than using the default VPC, you can define one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. For more information, see [How Amazon VPC works](#) in the *Amazon VPC User Guide*.

When you create an Auto Scaling group, you must choose the VPC and subnets where you will deploy the Auto Scaling group. Amazon EC2 Auto Scaling creates your instances in your chosen subnets. Each

instance is thus associated with a specific Availability Zone chosen by Amazon EC2 Auto Scaling. When instances launch, Amazon EC2 Auto Scaling tries to evenly distribute them between the zones for high availability and reliability.

The following image shows an overview of multi-tier architecture deployed across three Availability Zones.



Instance distribution

Amazon EC2 Auto Scaling automatically tries to maintain equivalent numbers of instances in each enabled Availability Zone. Amazon EC2 Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If there are multiple subnets chosen for the Availability Zone, Amazon EC2 Auto Scaling selects a subnet from the Availability Zone at random. If the attempt fails, however, Amazon EC2 Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds.

In circumstances where an Availability Zone becomes unhealthy or unavailable, the distribution of instances might become unevenly distributed across the Availability Zones. When the Availability Zone recovers, Amazon EC2 Auto Scaling automatically rebalances the Auto Scaling group. It does this by launching instances in the enabled Availability Zones with the fewest instances and terminating instances elsewhere.

Rebalancing activities

Rebalancing activities fall into two categories: Availability Zone rebalancing and capacity rebalancing.

Availability Zone rebalancing

After certain actions occur, your Auto Scaling group can become unbalanced between Availability Zones. Amazon EC2 Auto Scaling compensates by rebalancing the Availability Zones. The following actions can lead to rebalancing activity:

- You change the Availability Zones associated with your Auto Scaling group.
- You explicitly terminate or detach instances or place instances in standby, and then the group becomes unbalanced.
- An Availability Zone that previously had insufficient capacity recovers and now has additional capacity.
- An Availability Zone that previously had a Spot price above your maximum price now has a Spot price below your maximum price.

When rebalancing, Amazon EC2 Auto Scaling launches new instances before terminating the earlier ones. This way, rebalancing does not compromise the performance or availability of your application.

Because Amazon EC2 Auto Scaling attempts to launch new instances before terminating the earlier ones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities.

To avoid this problem, the system can temporarily exceed the specified maximum capacity of a group during a rebalancing activity. It can do so by a margin of 10 percent or one instance, whichever is greater. The margin is extended only if the group is at or near maximum capacity and needs rebalancing. This can happen because of user-requested rezoning, or to compensate for zone availability issues. The extension lasts only as long as needed to rebalance the group (typically a few minutes).

Capacity Rebalancing

You can turn on Capacity Rebalancing for your Auto Scaling groups when using Spot Instances. This lets Amazon EC2 Auto Scaling attempt to launch a Spot Instance whenever Amazon EC2 reports that a Spot Instance is at an elevated risk of interruption. After launching a new instance, it then terminates an earlier instance. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).

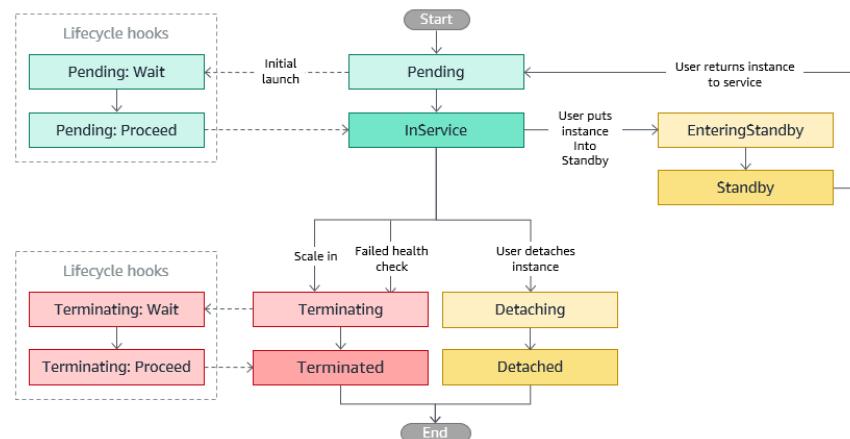
Amazon EC2 Auto Scaling instance lifecycle

The EC2 instances in an Auto Scaling group have a path, or lifecycle, that differs from that of other EC2 instances. The lifecycle starts when the Auto Scaling group launches an instance and puts it into service. The lifecycle ends when you terminate the instance, or the Auto Scaling group takes the instance out of service and terminates it.

Note

You are billed for instances as soon as they are launched, including the time that they are not yet in service.

The following illustration shows the transitions between instance states in the Amazon EC2 Auto Scaling lifecycle.



Scale out

The following scale-out events direct the Auto Scaling group to launch EC2 instances and attach them to the group:

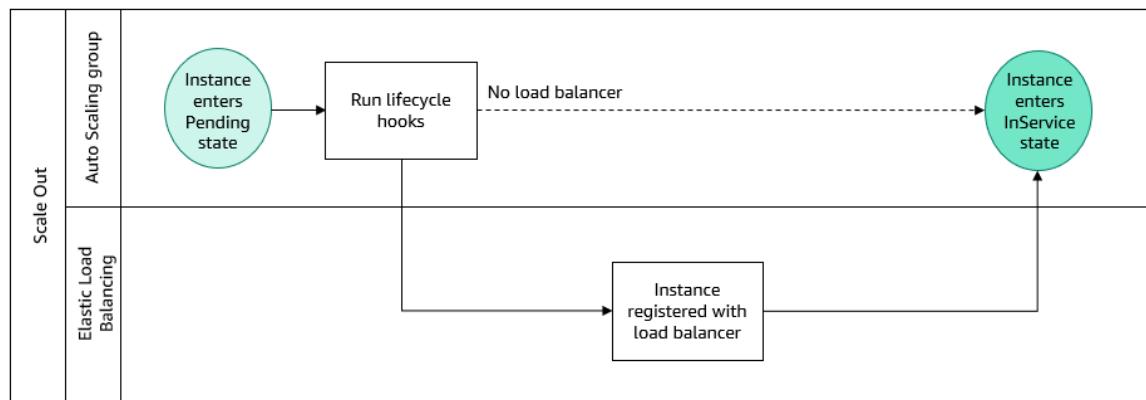
- You manually increase the size of the group. For more information, see [Manual scaling for Amazon EC2 Auto Scaling \(p. 124\)](#).
- You create a scaling policy to automatically increase the size of the group based on a specified increase in demand. For more information, see [Dynamic scaling for Amazon EC2 Auto Scaling \(p. 133\)](#).
- You set up scaling by schedule to increase the size of the group at a specific time. For more information, see [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#).

When a scale-out event occurs, the Auto Scaling group launches the required number of EC2 instances, using its assigned launch template. These instances start in the Pending state. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. For more information, see [Lifecycle hooks \(p. 10\)](#).

When each instance is fully configured and passes the Amazon EC2 health checks, it is attached to the Auto Scaling group and it enters the InService state. The instance is counted against the desired capacity of the Auto Scaling group.

If your Auto Scaling group is configured to receive traffic from an Elastic Load Balancing load balancer, Amazon EC2 Auto Scaling automatically registers your instance with the load balancer before it marks the instance as InService.

The following summarizes the workflow for registering an instance with a load balancer for a scale-out event.



Instances in service

Instances remain in the InService state until one of the following occurs:

- A scale-in event occurs, and Amazon EC2 Auto Scaling chooses to terminate this instance in order to reduce the size of the Auto Scaling group. For more information, see [Control which Auto Scaling instances terminate during scale in \(p. 230\)](#).
- You put the instance into a Standby state. For more information, see [Enter and exit standby \(p. 10\)](#).
- You detach the instance from the Auto Scaling group. For more information, see [Detach an instance \(p. 10\)](#).
- The instance fails a required number of health checks, so it is removed from the Auto Scaling group, terminated, and replaced. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).

Scale in

The following scale-in events direct the Auto Scaling group to detach EC2 instances from the group and terminate them:

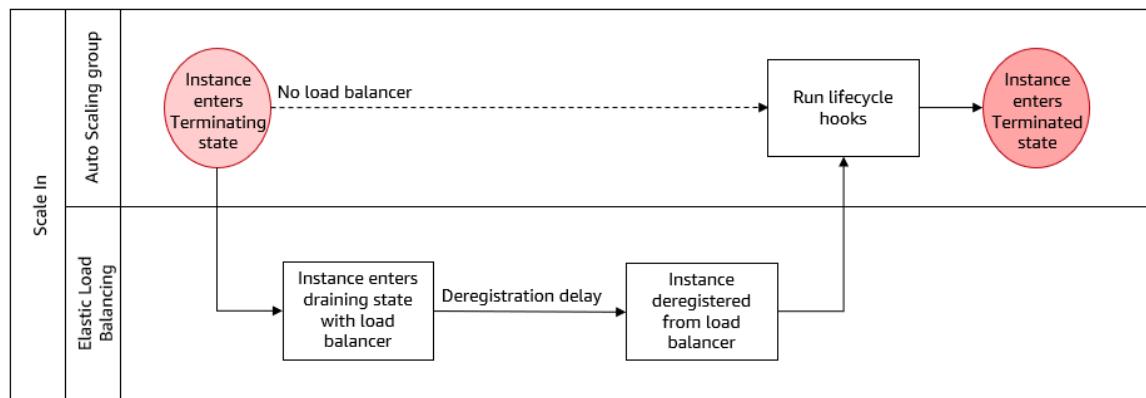
- You manually decrease the size of the group. For more information, see [Manual scaling for Amazon EC2 Auto Scaling \(p. 124\)](#).
- You create a scaling policy to automatically decrease the size of the group based on a specified decrease in demand. For more information, see [Dynamic scaling for Amazon EC2 Auto Scaling \(p. 133\)](#).
- You set up scaling by schedule to decrease the size of the group at a specific time. For more information, see [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#).

It is important that you create a corresponding scale-in event for each scale-out event that you create. This helps ensure that the resources assigned to your application match the demand for those resources as closely as possible.

When a scale-in event occurs, the Auto Scaling group terminates one or more instances. The Auto Scaling group uses its termination policy to determine which instances to terminate. Instances that are in the process of terminating from the Auto Scaling group and shutting down enter the Terminating state, and can't be put back into service. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. Finally, the instances are completely terminated and enter the Terminated state.

If your Auto Scaling group is configured to receive traffic from an Elastic Load Balancing load balancer, Amazon EC2 Auto Scaling automatically deregisters the terminating instance from the load balancer before running lifecycle hooks. Deregistering the instance ensures that all new requests are redirected to other instances in the load balancer's target group while existing connections to the instance are allowed to continue until the deregistration delay expires.

The following summarizes the workflow for deregistering an instance with a load balancer for a scale-in event.



Attach an instance

You can attach a running EC2 instance that meets certain criteria to your Auto Scaling group. After the instance is attached, it is managed as part of the Auto Scaling group.

For more information, see [Attach EC2 instances to your Auto Scaling group \(p. 126\)](#).

Detach an instance

You can detach an instance from your Auto Scaling group. After the instance is detached, you can manage it separately from the Auto Scaling group or attach it to a different Auto Scaling group.

For more information, see [Detach EC2 instances from your Auto Scaling group \(p. 130\)](#).

Lifecycle hooks

You can add a lifecycle hook to your Auto Scaling group so that you can perform custom actions when instances launch or terminate.

When Amazon EC2 Auto Scaling responds to a scale-out event, it launches one or more instances. These instances start in the Pending state. If you added an `autoscaling:EC2_INSTANCE_LAUNCHING` lifecycle hook to your Auto Scaling group, the instances move from the Pending state to the Pending:Wait state. After you complete the lifecycle action, the instances enter the Pending:Proceed state. When the instances are fully configured, they are attached to the Auto Scaling group and they enter the InService state.

When Amazon EC2 Auto Scaling responds to a scale-in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the Terminating state. If you added an `autoscaling:EC2_INSTANCE_TERMINATING` lifecycle hook to your Auto Scaling group, the instances move from the Terminating state to the Terminating:Wait state. After you complete the lifecycle action, the instances enter the Terminating:Proceed state. When the instances are fully terminated, they enter the Terminated state.

For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Enter and exit standby

You can put any instance that is in an InService state into a Standby state. This enables you to remove the instance from service, troubleshoot or make changes to it, and then put it back into service.

Instances in a Standby state continue to be managed by the Auto Scaling group. However, they are not an active part of your application until you put them back into service.

For more information, see [Temporarily remove instances from your Auto Scaling group \(p. 242\)](#).

Quotas for Amazon EC2 Auto Scaling

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for Amazon EC2 Auto Scaling, open the [Service Quotas console](#). In the navigation pane, choose **AWS services** and select **Amazon EC2 Auto Scaling**.

To request a quota increase, see [Requesting a Quota Increase](#) in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the [Auto Scaling Limits form](#). Quota increases are tied to the Region they were requested for.

All requests are submitted to AWS Support. You can track your request case in the AWS Support console.

Amazon EC2 Auto Scaling resources

Your AWS account has the following quotas related to the number of Auto Scaling groups and launch configurations that you can create.

Resource	Default quota
Auto Scaling groups per region	500
Launch configurations per region	200

Auto Scaling group configuration

Your AWS account has the following quotas related to the configuration of Auto Scaling groups. They cannot be changed.

Resource	Quota
Scaling policies per Auto Scaling group	50
Scheduled actions per Auto Scaling group	125
Step adjustments per step scaling policy	20
Lifecycle hooks per Auto Scaling group	50
SNS topics per Auto Scaling group	10
Classic Load Balancers per Auto Scaling group	50
Target groups per Auto Scaling group	50

Auto Scaling group API operations

Amazon EC2 Auto Scaling provides API operations to make changes to your Auto Scaling groups in batches. The following are the API limits on the maximum number of items (maximum array members) that are allowed in a single operation. They cannot be changed.

Operation	Maximum array members
AttachInstances	20 instance IDs
AttachLoadBalancers	10 load balancers
AttachLoadBalancerTargetGroups	10 target groups
BatchDeleteScheduledAction	50 scheduled actions
BatchPutScheduledUpdateGroupAction	50 scheduled actions
DetachInstances	20 instance IDs
DetachLoadBalancers	10 load balancers
DetachLoadBalancerTargetGroups	10 target groups
EnterStandby	20 instance IDs
ExitStandby	20 instance IDs

Operation	Maximum array members
SetInstanceProtection	50 instance IDs

Other services

Quotas for other services, such as Amazon EC2, can impact your Auto Scaling groups. For more information, see [Service endpoints and quotas](#) in the *Amazon Web Services General Reference*.

Set up Amazon EC2 Auto Scaling

Before you start using Amazon EC2 Auto Scaling, complete the following tasks.

Tasks

- [Prepare to use Amazon EC2 \(p. 13\)](#)
- [Install the AWS CLI \(p. 13\)](#)

Prepare to use Amazon EC2

If you haven't used Amazon EC2 before, complete the tasks described in the Amazon EC2 documentation. For more information, see [Setting up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances* or [Setting up with Amazon EC2](#) in the *Amazon EC2 User Guide for Windows Instances*.

Install the AWS CLI

To use the AWS CLI with Amazon EC2 Auto Scaling, install the latest AWS CLI version. For information about installing the AWS CLI or upgrading it to the latest version, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

To run AWS CLI commands using the AWS Management Console, you can use AWS CloudShell in supported AWS Regions. For more information, see [Create Auto Scaling groups from the command line using AWS CloudShell \(p. 283\)](#).

Get started with Amazon EC2 Auto Scaling

When you use Amazon EC2 Auto Scaling, you must use certain building blocks to get started. This tutorial walks you through the process for setting up building blocks to create a basic infrastructure for Amazon EC2 Auto Scaling.

Before you create an Auto Scaling group for use with your application, review your application thoroughly as it runs in the AWS Cloud. Consider the following:

- How many Availability Zones the Auto Scaling group should span.
- What existing resources can be used, such as security groups or Amazon Machine Images (AMIs).
- Whether you want to scale to increase or decrease capacity, or if you just want to ensure that a specific number of servers are always running. Keep in mind that Amazon EC2 Auto Scaling can do both simultaneously.
- What metrics have the most relevance to your application's performance.
- How long it takes to launch and configure a server.

The better you understand your application, the more effective you can make your Auto Scaling architecture.

Note

For an introduction video, see [AWS re:Invent 2018: Capacity Management Made Easy with Amazon EC2 Auto Scaling](#) on [YouTube](#).

Tasks

- [Walkthrough summary \(p. 14\)](#)
- [Prepare for the walkthrough \(p. 15\)](#)
- [Step 1: Create a launch template \(p. 15\)](#)
- [Step 2: Create a single-instance Auto Scaling group \(p. 17\)](#)
- [Step 3: Verify your Auto Scaling group \(p. 17\)](#)
- [Step 4: Terminate an instance in your Auto Scaling group \(p. 18\)](#)
- [Step 5: Next steps \(p. 18\)](#)
- [Step 6: Clean up \(p. 19\)](#)

Walkthrough summary

In this walkthrough, you:

- Create a configuration template that defines your EC2 instances. You can choose either the launch template or the launch configuration instructions. Although you can use a launch configuration, we recommend a launch template so that you can use the latest features of Amazon EC2 and Amazon EC2 Auto Scaling.
- Create an Auto Scaling group with a single instance in it.

- Terminate the instance and verify that the instance was removed from service and replaced. To maintain a constant number of instances, Amazon EC2 Auto Scaling detects and responds to Amazon EC2 health and reachability checks automatically.

If you created your AWS account less than 12 months ago, and have not already exceeded the [free tier](#) benefits for Amazon EC2, it will not cost you anything to complete this tutorial, because we help you select an instance type that is within the free tier benefits. Otherwise, when you follow this tutorial, you incur the standard Amazon EC2 usage fees from the time that the instance launches until you delete the Auto Scaling group (which is the final task of this tutorial) and the instance status changes to terminated.

Prepare for the walkthrough

This walkthrough assumes that you are familiar with launching EC2 instances and that you have already created a key pair and a security group. For more information, see [Setting up with Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you are new to Amazon EC2 Auto Scaling and want to get started using the service, you can use the *default VPC* for your AWS account. The default VPC includes a default public subnet in each Availability Zone and an internet gateway that is attached to your VPC. You can view your VPCs on the [Your VPCs page](#) of the Amazon Virtual Private Cloud (Amazon VPC) console.

Step 1: Create a launch template

In this step, you sign in to the Amazon EC2 console with your AWS account credentials and create a launch template that specifies the type of EC2 instance that Amazon EC2 Auto Scaling creates for you. Include information such as the ID of the Amazon Machine Image (AMI) to use, the instance type, the key pair, and security groups.

Note

Alternatively, you can use a launch configuration to create an Auto Scaling group instead of using a launch template. For the launch configuration instructions, see [Create a launch configuration](#).

To create a launch template

1. Open the [Launch templates page](#) of the Amazon EC2 console.
2. On the navigation bar at the top of the screen, select an AWS Region. The launch template and Auto Scaling group that you create are tied to the Region that you specify.
3. Choose **Create launch template**.
4. For **Launch template name**, enter **my-template-for-auto-scaling**.
5. Under **Auto Scaling guidance**, select the check box.
6. For **Application and OS Images (Amazon Machine Image)**, choose a version of Amazon Linux 2 (HVM) from the **Quick Start** list. The AMI serves as a basic configuration template for your instances.
7. For **Instance type**, choose a hardware configuration that is compatible with the AMI that you specified.

Note

If your account is less than 12 months old, you can use a **t2.micro** instance for free within certain usage limits. For more information, see [AWS free tier](#).

8. (Optional) For **Key pair (login)**, choose an existing key pair. You use key pairs to connect to an Amazon EC2 instance with SSH. Connecting to an instance is not included as part of this tutorial.

Therefore, you don't need to specify a key pair unless you intend to connect to your instance using SSH.

9. For **Network settings, Security groups**, choose a security group in the same VPC that you plan to use as the VPC for your Auto Scaling group. If you don't specify a security group, your instance is automatically associated with the default security group for the VPC.
10. You can leave **Advanced network configuration** empty. Leaving the setting empty creates a primary network interface with IP addresses that we select for your instance based on the subnet to which the network interface is established. If instead you choose to configure a network interface, the security group must be a part of it.
11. Choose **Create launch template**.
12. On the confirmation page, choose **Create Auto Scaling group**.

If you are not currently using launch templates and prefer not to create one now, you can create a launch configuration instead.

A launch configuration is similar to a launch template, in that it specifies the type of EC2 instance that Amazon EC2 Auto Scaling creates for you. You create the launch configuration by including information such as the ID of the Amazon Machine Image (AMI) to use, the instance type, the key pair, and security groups.

To create a launch configuration

1. Open the [Launch configurations page](#) of the Amazon EC2 console.
2. On the navigation bar, select an AWS Region. The launch configuration and Auto Scaling group that you create are tied to the Region that you specify.
3. Choose **Create launch configuration**, and then enter **my-first-launch-configuration** in the **Name** field.
4. For **Amazon machine image (AMI)**, choose an AMI. To find a specific AMI, you can [find a suitable AMI](#), make note of its ID, and enter the ID as search criteria.

To get the ID of the Amazon Linux 2 AMI:

- a. Open the [Amazon EC2 console](#).
- b. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch instances**.
- c. On the **Quick Start** tab of the [Choose an Amazon Machine Image](#) page, note the ID of the AMI next to **Amazon Linux 2 AMI (HVM)**. Notice that this AMI is marked "Free tier eligible."
5. For **Instance type**, select a hardware configuration for your instance.

Note
If your account is less than 12 months old, you can use a **t2.micro** instance for free within certain usage limits. For more information, see [AWS free tier](#).
6. Under **Additional configuration**, for **Advanced details, IP address type**, make a selection. To provide internet connectivity to instances in a VPC, choose an option that assigns a public IP address. If an instance is launched into a default VPC, the default is to assign a public IP address. If you want to provide internet connectivity to your instance but aren't sure whether you have a default VPC, choose **Assign a public IP address to every instance**.
7. For **Security groups**, choose an existing security group. If you leave the **Create a new security group** option selected, a default SSH rule is configured for Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.
8. For **Key pair (login)**, choose an option under **Key pair options** as instructed. Connecting to an instance is not included as part of this tutorial. Therefore, you can select **Proceed without a key pair** unless you intend to connect to your instance using SSH.
9. Choose **Create launch configuration**.

10. Select the check box next to the name of your new launch configuration and choose **Actions, Create Auto Scaling group**.

Step 2: Create a single-instance Auto Scaling group

Now use Amazon EC2 Auto Scaling to create an Auto Scaling group and add the launch template or launch configuration to the group. Also include information such as the VPC subnets for the instances.

Use the following procedure to continue where you left off after creating either a launch template or a launch configuration.

To create an Auto Scaling group

1. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter **my-first-asg**.
2. Choose **Next**.

The **Choose instance launch options** page appears, allowing you to choose the VPC network settings you want the Auto Scaling group to use and giving you options for launching On-Demand and Spot Instances (if you chose a launch template).

3. In the **Network** section, keep **VPC** set to the default VPC for your chosen AWS Region, or select your own VPC. The default VPC is automatically configured to provide internet connectivity to your instance. This VPC includes a public subnet in each Availability Zone in the Region.
4. For **Availability Zones and subnets**, choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).
5. [Launch template only] In the **Instance type requirements** section, use the default setting to simplify this step. (Do not override the launch template.) For this tutorial, you will launch only one On-Demand Instance using the instance type specified in your launch template.
6. Keep the rest of the defaults for this tutorial and choose **Skip to review**.

Note

The initial size of the group is determined by its desired capacity. The default value is 1 instance.

7. On the **Review** page, review the information for the group, and then choose **Create Auto Scaling group**.

Step 3: Verify your Auto Scaling group

Now that you have created an Auto Scaling group, you are ready to verify that the group has launched an EC2 instance.

Tip

In the following procedure, you look at the **Activity history** and **Instances** sections for the Auto Scaling group. In both, the named columns should already be displayed. To display hidden columns or change the number of rows shown, choose the gear icon on the top right corner of each section to open the preferences modal, update the settings as needed, and choose **Confirm**.

To verify that your Auto Scaling group has launched an EC2 instance

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.

2. Select the check box next to the Auto Scaling group that you just created.

A split pane opens up in the bottom of the **Auto Scaling groups** page. The first tab available is the **Details** tab, showing information about the Auto Scaling group.

3. Choose the second tab, **Activity**. Under **Activity history**, you can view the progress of activities that are associated with the Auto Scaling group. The **Status** column shows the current status of your instance. While your instance is launching, the status column shows **PreInService**. The status changes to **Successful** after the instance is launched. You can also use the refresh button to see the current status of your instance.
4. On the **Instance management** tab, under **Instances**, you can view the status of the instance.
5. Verify that your instance launched successfully. It takes a short time for an instance to launch.
 - The **Lifecycle** column shows the state of your instance. Initially, your instance is in the **Pending** state. After an instance is ready to receive traffic, its state is **InService**.
 - The **Health status** column shows the result of the EC2 instance health check on your instance.

Step 4: Terminate an instance in your Auto Scaling group

Use these steps to learn more about how Amazon EC2 Auto Scaling works, specifically, how it launches new instances when necessary. The minimum size for the Auto Scaling group that you created in this tutorial is one instance. Therefore, if you terminate that running instance, Amazon EC2 Auto Scaling must launch a new instance to replace it.

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.
3. On the **Instance management** tab, under **Instances**, select the ID of the instance.

This takes you to the **Instances** page of the Amazon EC2 console, where you can terminate the instance.

4. Choose **Actions, Instance State, Terminate**. When prompted for confirmation, choose **Yes, Terminate**.
5. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**. Select your Auto Scaling group and choose the **Activity** tab.

The default cooldown for the Auto Scaling group is 300 seconds (5 minutes), so it takes about 5 minutes until you see the scaling activity. In the activity history, when the scaling activity starts, you see an entry for the termination of the first instance and an entry for the launch of a new instance.

6. On the **Instance management** tab, the **Instances** section shows the new instance only.
7. On the navigation pane, under **Instances**, choose **Instances**. This page shows both the terminated instance and the new running instance.

Step 5: Next steps

Go to the next step if you would like to delete the basic infrastructure for automatic scaling that you just created. Otherwise, you can use this infrastructure as your base and try one or more of the following:

- Connect to your Linux instance using Session Manager or SSH. For more information, see [Connect to your Linux instance using Session Manager](#) and [Connect to your Linux instance using SSH](#) in the [Amazon EC2 User Guide for Linux Instances](#).

- Configure an Amazon SNS notification to notify you whenever your Auto Scaling group launches or terminates instances. For more information, see [Monitor with Amazon SNS notifications \(p. 271\)](#).
- Manually scale your Auto Scaling group to test the SNS notification. For more information, see [Manual scaling \(p. 124\)](#).

You can also start familiarizing yourself with auto scaling concepts by reading about [Target tracking scaling policies \(p. 135\)](#). If the load on your application changes, your Auto Scaling group can scale out (add instances) and scale in (run fewer instances) automatically by adjusting the desired capacity of the group between the minimum and maximum capacity limits. For more information about setting these limits, see [Set capacity limits on your Auto Scaling group \(p. 122\)](#).

If you plan to attach a load balancer to your Auto Scaling group, you can learn how to create one quickly using the Amazon EC2 Auto Scaling console. For more information, see [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#).

Step 6: Clean up

You can either delete your scaling infrastructure or delete just your Auto Scaling group and keep your launch template or launch configuration to use later.

If you launched an instance that is not within the [AWS Free Tier](#), you should terminate your instance to prevent additional charges. When you terminate the instance, the data associated with it will also be deleted.

To delete your Auto Scaling group

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group (my-first-asg).
3. Choose **Delete**.
4. When prompted for confirmation, type **delete** to confirm deleting the specified Auto Scaling group and then choose **Delete**.

A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. When the deletion has occurred, the **Desired**, **Min**, and **Max** columns show 0 instances for the Auto Scaling group. It takes a few minutes to terminate the instance and delete the group. Refresh the list to see the current state.

Skip the following procedure if you would like to keep your launch template.

To delete your launch template

1. Open the [Launch templates page](#) of the Amazon EC2 console.
2. Select your launch template (my-template-for-auto-scaling).
3. Choose **Actions, Delete template**.
4. When prompted for confirmation, type **Delete** to confirm deleting the specified launch template and then choose **Delete**.

Skip the following procedure if you would like to keep your launch configuration.

To delete your launch configuration

1. Open the [Launch configurations page](#) of the Amazon EC2 console.

2. Select your launch configuration (my-first-launch-configuration).
3. Choose **Actions, Delete launch configuration**.
4. When prompted for confirmation, choose **Delete**.

Launch templates

A launch template is similar to a [launch configuration \(p. 40\)](#), in that it specifies instance configuration information. It includes the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and other parameters used to launch EC2 instances. However, defining a launch template instead of a launch configuration allows you to have multiple versions of a launch template.

With versioning of launch templates, you can create a subset of the full set of parameters. Then, you can reuse it to create other versions of the same launch template. For example, you can create a launch template that defines a base configuration without an AMI or user data script. After you create your launch template, you can create a new version and add the AMI and user data that has the latest version of your application for testing. This results in two versions of the launch template. Storing a base configuration helps you to maintain the required general configuration parameters. You can create a new version of your launch template from the base configuration whenever you want. You can also delete the versions used for testing your application when you no longer need them.

We recommend that you use launch templates to ensure that you're accessing the latest features and improvements. Not all Amazon EC2 Auto Scaling features are available when you use launch configurations. For example, you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types. You must use a launch template to configure these features. For more information, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

With launch templates, you can also use newer features of Amazon EC2. This includes the current generation of EBS Provisioned IOPS volumes (io2), EBS volume tagging, [T2 Unlimited instances](#), Elastic Inference, and [Dedicated Hosts](#), to name a few. Dedicated Hosts are physical servers with EC2 instance capacity that are dedicated to your use. While Amazon EC2 [Dedicated Instances](#) also run on dedicated hardware, the advantage of using Dedicated Hosts over Dedicated Instances is that you can bring eligible software licenses from external vendors and use them on EC2 instances.

If you currently use launch configurations, you can migrate data from your existing launch configurations to launch templates by [copying them in the console](#). Then, you can migrate your deployed Auto Scaling groups that use a launch configuration to a new launch template. To do this, start an instance refresh to do a rolling update of your group. For more information, see [Replace Auto Scaling instances \(p. 99\)](#).

Note

For more information about migrating to launch templates, see [Amazon EC2 Auto Scaling will no longer add support for new EC2 features to Launch Configurations](#) on the AWS Compute Blog.

When you create a launch template, all parameters are optional. However, if a launch template does not specify an AMI, you cannot add the AMI when you create your Auto Scaling group. If you specify an AMI but no instance type, you can add one or more instance types when you create your Auto Scaling group.

Contents

- [Permissions \(p. 22\)](#)
- [Create a launch template for an Auto Scaling group \(p. 22\)](#)
- [Copy launch configurations to launch templates \(p. 30\)](#)
- [Replace a launch configuration with a launch template \(p. 31\)](#)
- [Request Spot Instances for fault-tolerant and flexible applications \(p. 32\)](#)

- [Examples for creating and managing launch templates with the AWS Command Line Interface \(AWS CLI\) \(p. 33\)](#)

Permissions

The procedures in this section assume that you already have the required permissions to use launch templates. With permissions in place, you can create and manage launch templates. You can also create and update Auto Scaling groups and specify a launch template instead of a launch configuration.

When you update or create an Auto Scaling group and specify a launch template, your `ec2:RunInstances` permissions are checked. If you do not have sufficient permissions, you receive an error that you're not authorized to use the launch template.

Some additional functionality in the request requires additional permissions, such as the ability to pass an IAM role to provisioned instances or to add tags to provisioned instances and volumes.

For information about how an administrator grants you permissions, see [Launch template support \(p. 347\)](#).

Create a launch template for an Auto Scaling group

Before you can create an Auto Scaling group using a launch template, you must create a launch template with the parameters required to launch an EC2 instance. These parameters include the ID of the Amazon Machine Image (AMI) and an instance type.

A launch template provides full functionality for Amazon EC2 Auto Scaling and also newer features of Amazon EC2 such as the current generation of Amazon EBS Provisioned IOPS volumes (io2), EBS volume tagging, T2 Unlimited instances, Elastic Inference, and Dedicated Hosts.

To create new launch templates, use the following procedures.

Contents

- [Create your launch template \(console\) \(p. 23\)](#)
 - [Change the default network interface settings \(p. 24\)](#)
 - [Modify the storage configuration \(p. 25\)](#)
 - [Configure advanced settings for your launch template \(p. 27\)](#)
- [Create a launch template from an existing instance \(console\) \(p. 29\)](#)
- [Additional information \(p. 29\)](#)
- [Limitations \(p. 30\)](#)

Important

Launch template parameters are not fully validated when you create the launch template. If you specify incorrect values for parameters, or if you do not use supported parameter combinations, no instances can launch using this launch template. Be sure to specify the correct values for the parameters and use supported parameter combinations. For example, to launch instances with an Arm-based AWS Graviton or Graviton2 AMI, you must specify an Arm-compatible instance type.

Create your launch template (console)

The following steps describe how to configure your launch template:

- Specify the Amazon Machine Image (AMI) from which to launch the instances.
- Choose an instance type that is compatible with the AMI that you specify.
- Specify the key pair to use when connecting to instances, for example, using SSH.
- Add one or more security groups to allow relevant access to the instances from an external network.
- Specify whether to attach additional volumes to each instance.
- Add custom tags (key-value pairs) to the instances and volumes.

To create a launch template

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Instances**, choose **Launch Templates**.
3. Choose **Create launch template**. Enter a name and provide a description for the initial version of the launch template.
4. Under **Auto Scaling guidance**, select the check box to have Amazon EC2 provide guidance to help create a template to use with Amazon EC2 Auto Scaling.
5. Under **Launch template contents**, fill out each required field and any optional fields as needed.
 - a. **Application and OS Images (Amazon Machine Image)**: (Required) Choose the ID of the AMI for your instances. You can search through all available AMIs, or select an AMI from the **Recents** or **Quick Start** list. If you don't see the AMI that you need, choose **Browse more AMIs** to browse the full AMI catalog.

To choose a custom AMI, you must first create your AMI from a customized instance. For more information, see [Create an AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

- b. For **Instance type**, choose a single instance type that's compatible with the AMI that you specified.

Alternatively, to launch an Auto Scaling group with multiple instance types, choose **Advanced, Specify instance type attributes**, and then specify the following options:

- **Number of vCPUs**: Enter the minimum and maximum number of vCPUs. To indicate no limits, enter a minimum of 0, and keep the maximum blank.
- **Amount of memory (MiB)**: Enter the minimum and maximum amount of memory, in MiB. To indicate no limits, enter a minimum of 0, and keep the maximum blank.
- Expand **Optional instance type attributes** and choose **Add attribute** to further limit the types of instances that can be used to fulfill your desired capacity. For information about each attribute, see [InstanceRequirementsRequest](#) in the *Amazon EC2 API Reference*.
- **Resulting instance types**: You can view the instance types that match the specified compute requirements, such as vCPUs, memory, and storage.
- To exclude instance types, choose **Add attribute**. From the **Attribute** list, choose **Excluded instance types**. From the **Attribute value** list, select the instance types to exclude.

For more information, see [Create an Auto Scaling group using attribute-based instance type selection \(p. 75\)](#).

- c. **Key pair (login)**: For **Key pair name**, choose an existing key pair, or choose **Create new key pair** to create a new one. For more information, see [Amazon EC2 key pairs and Linux instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

- d. **Network settings:** For **Firewall (security groups)**, use one or more security groups, or keep this blank and configure one or more security groups as part of the network interface. For more information, see [Amazon EC2 security groups for Linux instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you don't specify any security groups in your launch template, Amazon EC2 uses the default security group for the VPC that your Auto Scaling group will launch instances into. By default, this security group doesn't allow inbound traffic from external networks. For more information, see [Default security groups for your VPCs](#) in the *Amazon VPC User Guide*.

- e. Do one of the following:
 - Change the default network interface settings. For example, you can enable or disable the public IPv4 addressing feature, which overrides the auto-assign public IPv4 addresses setting on the subnet. For more information, see [Change the default network interface settings \(p. 24\)](#).
 - Skip this step to keep the default network interface settings.
 - f. Do one of the following:
 - Modify the storage configuration. For more information, see [Modify the storage configuration \(p. 25\)](#).
 - Skip this step to keep the default storage configuration.
 - g. For **Resource tags**, specify tags by providing key and value combinations. If you specify instance tags in your launch template and then you choose to propagate your Auto Scaling group's tags to its instances, all the tags are merged. If the same tag key is specified for a tag in your launch template and a tag in your Auto Scaling group, then the tag value from the group takes precedence.
6. (Optional) Configure advanced settings. For more information, see [Configure advanced settings for your launch template \(p. 27\)](#).
 7. When you are ready to create the launch template, choose **Create launch template**.
 8. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Change the default network interface settings

This section shows you how to change the default network interface settings. For example, you can define whether you want to assign a public IPv4 address to each instance instead of defaulting to the auto-assign public IPv4 addresses setting on the subnet.

Considerations and limitations

When changing the default network interface settings, keep in mind the following considerations and limitations:

- You must configure the security groups as part of the network interface, not in the **Security groups** section of the template. You cannot specify security groups in both places.
- You cannot assign secondary private IP addresses, known as *secondary IP addresses*, to a network interface.
- If you specify an existing network interface ID, you can launch only one instance. To do this, you must use the AWS CLI or an SDK to create the Auto Scaling group. When you create the group, you must specify the Availability Zone, but not the subnet ID. Also, you can specify an existing network interface only if it has a device index of 0.
- You cannot auto-assign a public IPv4 address if you specify more than one network interface. You also cannot specify duplicate device indexes across network interfaces. Both the primary and secondary network interfaces reside in the same subnet. For more information, see [Provide network connectivity for your Auto Scaling instances using Amazon VPC \(p. 317\)](#).

- When an instance launches, a private address is automatically allocated to each network interface. The address comes from the CIDR range of the subnet in which the instance is launched. For information on specifying CIDR blocks (or IP address ranges) for your VPC or subnet, see the [Amazon VPC User Guide](#).

To change the default network interface settings

1. Under **Network settings**, expand **Advanced network configuration**.
2. Choose **Add network interface** to configure the primary network interface, paying attention to the following fields:
 - a. **Device index**: Keep the default value, 0, to apply your changes to the primary network interface (eth0).
 - b. **Network interface**: Keep the default value, **New interface**, to have Amazon EC2 Auto Scaling automatically create a new network interface when an instance is launched. Alternatively, you can choose an existing, available network interface with a device index of 0, but this limits your Auto Scaling group to one instance.
 - c. **Description**: (Optional) Enter a descriptive name.
 - d. **Subnet**: Keep the default **Don't include in launch template** setting.

If the AMI specifies a subnet for the network interface, this results in an error. We recommend turning off **Auto Scaling guidance** as a workaround. After you make this change, you will not receive an error message. However, regardless of where the subnet is specified, the subnet settings of the Auto Scaling group take precedence and cannot be overridden.
 - e. **Auto-assign public IP**: Change whether your network interface with a device index of 0 receives a public IPv4 address. By default, instances in a default subnet receive a public IPv4 address, while instances in a nondefault subnet do not. Select **Enable** or **Disable** to override the subnet's default setting.
 - f. **Security groups**: Choose one or more security groups for the network interface. Each security group must be configured for the VPC that your Auto Scaling group will launch instances into. For more information, see [Amazon EC2 security groups for Linux instances](#) in the [Amazon EC2 User Guide for Linux Instances](#).
 - g. **Delete on termination**: Choose **Yes** to delete the network interface when the instance is terminated, or choose **No** to keep the network interface.
 - h. **Elastic Fabric Adapter**: To support high performance computing (HPC) use cases, change the network interface into an Elastic Fabric Adapter network interface. For more information, see [Elastic Fabric Adapter](#) in the [Amazon EC2 User Guide for Linux Instances](#).
 - i. **Network card index**: Choose **0** to attach the primary network interface to the network card with a device index of 0. If this option isn't available, keep the default value, **Don't include in launch template**. Attaching the network interface to a specific network card is available only for supported instance types. For more information, see [Network cards](#) in the [Amazon EC2 User Guide for Linux Instances](#).
3. To add a secondary network interface, choose **Add network interface**.

Modify the storage configuration

You can modify the storage configuration for instances launched from an Amazon EBS-backed AMI or an instance store-backed AMI. You can also specify additional EBS volumes to attach to the instances. The AMI includes one or more volumes of storage, including the root volume (**Volume 1 (AMI Root)**).

To modify the storage configuration

1. In **Configure storage**, modify the size or type of volume.

If the value you specify for volume size is outside the limits of the volume type, or smaller than the snapshot size, an error message is displayed. To help you address the issue, this message gives the minimum or maximum value that the field can accept.

Only volumes associated with an Amazon EBS-backed AMI appear. To display information about the storage configuration for an instance launched from an instance store-backed AMI, choose **Show details** from the **Instance store volumes** section.

To specify all EBS volume parameters, switch to the **Advanced** view in the top right corner.

2. For advanced options, expand the volume that you want to modify and configure the volume as follows:
 - a. **Storage type:** The type of volume (EBS or ephemeral) to associate with your instance. The instance store (ephemeral) volume type is only available if you select an instance type that supports it. For more information, see [Amazon EC2 instance store](#) and [Amazon EBS volumes](#) in the *Amazon EC2 User Guide for Linux Instances*.
 - b. **Device name:** Select from the list of available device names for the volume.
 - c. **Snapshot:** Select the snapshot from which to create the volume. You can search for available shared and public snapshots by entering text into the **Snapshot** field.
 - d. **Size (GiB):** For EBS volumes, you can specify a storage size. If you have selected an AMI and instance that are eligible for the free tier, keep in mind that to stay within the free tier, you must stay under 30 GiB of total storage. For more information, see [Constraints on the size and configuration of an EBS volume](#) in the *Amazon EC2 User Guide for Linux Instances*.
 - e. **Volume type:** For EBS volumes, choose the volume type. For more information, see [Amazon EBS volume types](#) in the *Amazon EC2 User Guide for Linux Instances*.
 - f. **IOPS:** If you have selected a Provisioned IOPS SSD (io1 and io2) or General Purpose SSD (gp3) volume type, then you can enter the number of I/O operations per second (IOPS) that the volume can support. This is required for io1, io2, and gp3 volumes. It is not supported for gp2, st1, sc1, or standard volumes.
 - g. **Delete on termination:** For EBS volumes, choose **Yes** to delete the volume when the instance is terminated, or choose **No** to keep the volume.
 - h. **Encrypted:** If the instance type supports EBS encryption, you can choose **Yes** to enable encryption for the volume. If you have enabled encryption by default in this Region, encryption is enabled for you. For more information, see [Amazon EBS encryption](#) and [Encryption by default](#) in the *Amazon EC2 User Guide for Linux Instances*.

The default effect of setting this parameter varies with the choice of volume source, as described in the following table. In all cases, you must have permission to use the specified AWS KMS key.

Encryption outcomes

If Encrypted parameter is set to...	And if source of volume is...	Then the default encryption state is...	Notes
No	New (empty) volume	Unencrypted*	N/A
	Unencrypted snapshot that you own	Unencrypted*	
	Encrypted snapshot that you own	Encrypted by same key	

If Encrypted parameter is set to...	And if source of volume is...	Then the default encryption state is...	Notes
	Unencrypted snapshot that is shared with you	Unencrypted*	
	Encrypted snapshot that is shared with you	Encrypted by default KMS key	
Yes	New volume	Encrypted by default KMS key	To use a non-default KMS key, specify a value for the KMS key parameter.
	Unencrypted snapshot that you own	Encrypted by default KMS key	
	Encrypted snapshot that you own	Encrypted by same key	
	Unencrypted snapshot that is shared with you	Encrypted by default KMS key	
	Encrypted snapshot that is shared with you	Encrypted by default KMS key	

* If encryption by default is enabled, all newly created volumes (whether or not the **Encrypted** parameter is set to **Yes**) are encrypted using the default KMS key. If you set both the **Encrypted** and **KMS key** parameters, then you can specify a non-default KMS key.

- i. **KMS key:** If you chose **Yes** for **Encrypted**, then you must select a customer managed key to use to encrypt the volume. If you have enabled encryption by default in this Region, the default customer managed key is selected for you. You can select a different key or specify the ARN of any customer managed key that you previously created using the AWS Key Management Service.
3. To specify additional volumes to attach to the instances launched by this launch template, choose **Add new volume**.

Configure advanced settings for your launch template

You can define any additional capabilities that your Auto Scaling instances need. For example, you can choose an IAM role that your application can use when it accesses other AWS resources or specify the instance user data that can be used to perform common automated configuration tasks after an instance starts.

The following steps discuss the most useful settings to pay attention to. For more information about any of the settings under **Advanced details**, see [Creating a launch template](#) in the *Amazon EC2 User Guide for Linux Instances*.

To configure advanced settings

1. For **Advanced details**, expand the section to view the fields.
2. For **Purchasing option**, you can choose **Request Spot Instances** to request Spot Instances at the Spot price, capped at the On-Demand price, and choose **Customize** to change the default Spot Instance settings. For an Auto Scaling group, you must specify a one-time request with no end date (the default). For more information, see [Request Spot Instances for fault-tolerant and flexible applications \(p. 32\)](#).

Note

Amazon EC2 Auto Scaling lets you override the instance type in your launch template to create an Auto Scaling group that uses multiple instance types and launches Spot and On-Demand Instances. To do so, you must leave **Purchasing option** unspecified in your launch template.

If you try to create a mixed instances group using a launch template with **Purchasing option** specified, you get the following error.

Incompatible launch template: You cannot use a launch template that is set to request Spot Instances (InstanceMarketOptions) when you configure an Auto Scaling group with a mixed instances policy. Add a different launch template to the group and try again.

For information about creating mixed instances groups, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

3. For **IAM instance profile**, you can specify an AWS Identity and Access Management (IAM) instance profile to associate with the instances. When you choose an instance profile, you associate the corresponding IAM role with the EC2 instances. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).
4. For **Termination protection**, choose whether to protect instances from accidental termination. When you enable termination protection, it provides additional termination protection, but it does not protect from Amazon EC2 Auto Scaling initiated termination. To control whether an Auto Scaling group can terminate a particular instance, use [Use instance scale-in protection \(p. 240\)](#).
5. For **Detailed CloudWatch monitoring**, choose whether to enable the instances to publish metric data at 1-minute intervals to Amazon CloudWatch. Additional charges apply. For more information, see [Configure monitoring for Auto Scaling instances \(p. 265\)](#).
6. For **Elastic inference**, choose an elastic inference accelerator to attach to your EC2 CPU instance. Additional charges apply. For more information, see [Working with Amazon Elastic Inference](#) in the [Amazon Elastic Inference Developer Guide](#).
7. For **T2/T3 Unlimited**, choose whether to enable applications to burst beyond the baseline for as long as needed. This field is only valid for T2, T3, and T3a instances. Additional charges may apply. For more information, see [Using an Auto Scaling group to launch a burstable performance instance as Unlimited](#) in the [Amazon EC2 User Guide for Linux Instances](#).
8. For **Placement group name**, you can specify a placement group in which to launch the instances. Not all instance types can be launched in a placement group. If you configure an Auto Scaling group using a CLI command that specifies a different placement group, the placement group for the Auto Scaling group takes precedence.
9. For **Capacity Reservation**, you can specify whether to launch the instances into shared capacity, any open Capacity Reservation, a specific Capacity Reservation, or a Capacity Reservation group. For more information, see [Launching instances into an existing capacity reservation](#) in the [Amazon EC2 User Guide for Linux Instances](#).
10. For **Tenancy**, you can choose to run your instances on shared hardware (**Shared**), on dedicated hardware (**Dedicated**), or when using a host resource group, on Dedicated Hosts (**Dedicated host**). Additional charges may apply.

If you chose **Dedicated Hosts**, complete the following information:

- For **Tenancy host resource group**, you can specify a host resource group for a BYOL AMI to use on Dedicated Hosts. You do not have to have already allocated Dedicated Hosts in your account before you use this feature. Your instances will automatically launch onto Dedicated Hosts regardless. Note that an AMI based on a license configuration association can be mapped to only one host resource group at a time. For more information, see [Host resource groups](#) in the [AWS License Manager User Guide](#).
11. For **License configurations**, specify the license configuration to use. You can launch instances against the specified license configuration to track your license usage. For more information, see [Create a license configuration](#) in the [License Manager User Guide](#).

12. To configure instance metadata options for all of the instances that are associated with this version of the launch template, do the following:
 - a. For **Metadata accessible**, choose whether to enable or disable access to the HTTP endpoint of the instance metadata service. By default, the HTTP endpoint is enabled. If you choose to disable the endpoint, access to your instance metadata is turned off. You can specify the condition to require IMDSv2 only when the HTTP endpoint is enabled.
 - b. For **Metadata version**, you can choose to require the use of Instance Metadata Service Version 2 (IMDSv2) when requesting instance metadata. If you do not specify a value, the default is to support both IMDSv1 and IMDSv2.
 - c. For **Metadata token response hop limit**, you can set the allowable number of network hops for the metadata token. If you do not specify a value, the default is 1.

For more information, see [Configuring the instance metadata service](#) in the *Amazon EC2 User Guide for Linux Instances*.

13. For **User data**, you can add shell scripts and cloud-init directives to customize an instance at launch. For more information, see [Run commands on your Linux instance at launch](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

Running scripts at launch adds to the amount of time it takes for an instance to be ready for use. However, you can allow extra time for the scripts to complete before the instance enters the InService state by adding a lifecycle hook to the Auto Scaling group. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

14. Choose **Create launch template**.
15. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Create a launch template from an existing instance (console)

To create a launch template from an existing instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Instances**, choose **Instances**.
3. Select the instance and choose **Actions, Image and templates, Create template from instance**.
4. Provide a name and description.
5. Under **Auto Scaling guidance**, select the check box.
6. Adjust any settings as required, and choose **Create launch template**.
7. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Additional information

For additional information about creating launch templates, see:

- [Launching an instance from a launch template](#) section of the *Amazon EC2 User Guide for Linux Instances*
- [Auto scaling template snippets](#) section of the *AWS CloudFormation User Guide*
- [AWS::EC2::LaunchTemplate](#) section of the *AWS CloudFormation User Guide*

For the procedures for creating an Auto Scaling group with a launch template, see the following topics:

- [Create an Auto Scaling group using a launch template \(p. 81\)](#)
- [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#)
- [Create an Auto Scaling group using attribute-based instance type selection \(p. 75\)](#)

Limitations

- Amazon EC2 lets you configure a subnet in a launch template. However, the subnet settings of the Auto Scaling group take precedence over the subnet settings of the launch template.
- Because the subnet settings in your launch template are ignored in favor of what is specified in the Auto Scaling group, all of the network interfaces that are created for a given instance will be connected to the same subnet as the instance. For additional limitations on user-defined network interfaces, see [Change the default network interface settings \(p. 24\)](#).
- A launch template lets you configure additional settings in your Auto Scaling group to launch multiple instance types and combine On-Demand and Spot purchase options, as described in [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#). Launching instances with such a combination is not supported if you specify a Spot Instance request in the launch template.
- Support for Dedicated Hosts (host tenancy) is only available if you specify a host resource group. You cannot target a specific host ID or use host placement affinity.

Copy launch configurations to launch templates

To migrate from launch configurations to launch templates, you must copy or recreate your launch configurations as launch templates. We recommend that you migrate to launch templates to take advantage of the latest features of Amazon EC2 and Amazon EC2 Auto Scaling.

If you copy your launch configurations, you can migrate them all at once, or you can perform an incremental migration over time by choosing which launch configurations to copy. The copying feature is available only from the console.

To copy a launch configuration to a launch template (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
3. Select the launch configuration you want to copy and choose **Copy to launch template, Copy selected**. This sets up a new launch template with the same name and options as the launch configuration that you selected.
4. For **New launch template name**, you can use the name of the launch configuration (the default) or enter a new name. Launch template names must be unique.
5. (Optional) To create an Auto Scaling group using the new launch template, select **Create an Auto Scaling group using the new template**.
6. Choose **Copy**.

If you know that you want to copy all launch configurations to launch templates, use the following procedure.

To copy all launch configurations to launch templates (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.

3. Choose **Copy to launch template, Copy all**. This copies each launch configuration in the current Region to a new launch template with the same name and options.
4. Choose **Copy**.

Next, you can update your existing Auto Scaling groups to specify the launch templates that you created. For more information, see [Replace a launch configuration with a launch template \(p. 31\)](#). As another option, you can follow the procedure in [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#) to add the new launch templates to your Auto Scaling groups and update your Auto Scaling instances immediately.

Replace a launch configuration with a launch template

When you edit an Auto Scaling group that has an existing launch configuration, you have the option of replacing the launch configuration with a launch template. This lets you use launch templates with any Auto Scaling groups that you currently use. In doing so, you can take advantage of versioning and other features of launch templates.

After you replace the launch configuration for an Auto Scaling group, any new instances are launched using the new launch template. Existing instances are not affected. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow automatic scaling to gradually replace earlier instances with newer instances based on your [termination policies \(p. 230\)](#).

Note

With the instance refresh feature, you can replace the instances in the Auto Scaling group to launch new instances that use the launch template immediately. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Prerequisites

Before you can replace a launch configuration in an Auto Scaling group, you must first create your launch template. A basic way to create a launch template is to copy it from the launch configuration. For more information, see [Copy launch configurations to launch templates \(p. 30\)](#).

If you switch your Auto Scaling group from using a launch configuration, be sure that your permissions are up to date. In order to use a launch template, you need specific [permissions](#).

To replace the launch configuration for an Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom part of the page, showing information about the group that's selected.

3. On the **Details** tab, choose **Launch configuration, Edit**.
4. Choose **Switch to launch template**.
5. For **Launch template**, select your launch template.
6. For **Version**, select the launch template version, as needed. After you create versions of a launch template, you can choose whether the Auto Scaling group uses the default or the latest version of the launch template when scaling out.
7. When you have finished, choose **Update**.

To replace a launch configuration using the command line

You can use one of the following commands:

- [update-auto-scaling-group](#) (AWS CLI)
- [Update-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

For examples of using a CLI command to update an Auto Scaling group to use a launch template, see [Update an Auto Scaling group to use a launch template \(p. 39\)](#).

Request Spot Instances for fault-tolerant and flexible applications

In your launch template, you can optionally request Spot Instances with no end date or duration. Amazon EC2 Spot Instances are spare capacity available at steep discounts compared to the EC2 On-Demand price. Spot Instances are a cost-effective choice if you can be flexible about when your applications run and if your applications can be interrupted. For more information about creating a launch template that requests Spot Instances, see [Configure advanced settings for your launch template \(p. 27\)](#).

Important

Spot Instances are typically used to supplement On-Demand Instances. For this scenario, you can specify the same settings that are used to launch Spot Instances as part of the settings of your Auto Scaling group. When you specify the settings as part of the Auto Scaling group, you can request to launch Spot Instances only after launching a certain number of On-Demand Instances and then continue to launch some combination of On-Demand Instances and Spot Instances as the group scales. For more information, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

This topic describes how to launch only Spot Instances in your Auto Scaling group by specifying settings in a launch template, rather than in the Auto Scaling group itself. The information in this topic also applies to Auto Scaling groups that request Spot Instances with a [launch configuration \(p. 40\)](#). The difference is that a launch configuration requires a maximum price, but for launch templates, the maximum price is optional.

When you create a launch template to launch only Spot Instances, keep the following considerations in mind:

- **Spot price.** You pay only the current Spot price for the Spot Instances that you launch. This pricing changes slowly over time based on long-term trends in supply and demand. For more information, see [Spot Instances](#) and [Pricing and savings](#) in the *Amazon EC2 User Guide for Linux Instances*.
- **Setting your maximum price.** You can optionally include a maximum price per hour for Spot Instances in your launch template. If your maximum price exceeds the current Spot price, the Amazon EC2 Spot service fulfills your request immediately if capacity is available. If the price for Spot Instances rises above your maximum price for a running instance in your Auto Scaling group, it terminates your instance.

Warning

Your application might not run if you do not receive any Spot Instances, such as when your maximum price is too low. To take advantage of the Spot Instances available for as long as possible, set your maximum price close to the On-Demand price.

- **Balancing across Availability Zones.** If you specify multiple Availability Zones, Amazon EC2 Auto Scaling distributes the Spot requests across the specified zones. If your maximum price is too low in one Availability Zone for any requests to be fulfilled, Amazon EC2 Auto Scaling checks whether

requests were fulfilled in the other zones. If so, Amazon EC2 Auto Scaling cancels the requests that failed and redistributes them across the Availability Zones that have requests fulfilled. If the price in an Availability Zone with no fulfilled requests drops enough that future requests succeed, Amazon EC2 Auto Scaling rebalances across all of the Availability Zones.

- **Spot Instance termination.** Spot Instances can be terminated at any time. The Amazon EC2 Spot service can terminate Spot Instances in your Auto Scaling group as the availability of, or price for, Spot Instances changes. When scaling or performing health checks, Amazon EC2 Auto Scaling can also terminate Spot Instances in the same way that it can terminate On-Demand Instances. When an instance is terminated, any storage is deleted.
- **Maintaining your desired capacity.** When a Spot Instance is terminated, Amazon EC2 Auto Scaling attempts to launch another Spot Instance to maintain the desired capacity for the group. If the current Spot price is less than your maximum price, it launches a Spot Instance. If the request for a Spot Instance is unsuccessful, it keeps trying.
- **Changing your maximum price.** To change your maximum price, create a new launch template or update an existing launch template with the new maximum price, and then associate it with your Auto Scaling group. The existing Spot Instances continue to run as long as the maximum price specified in the launch template used for those instances is higher than the current Spot price. If you did not set a maximum price, the default maximum price is the On-Demand price.

Examples for creating and managing launch templates with the AWS Command Line Interface (AWS CLI)

You can create and manage launch templates through the AWS Management Console, AWS CLI, or SDKs. This section shows you examples of creating and managing launch templates for Amazon EC2 Auto Scaling from the AWS CLI.

Contents

- [Example usage \(p. 33\)](#)
- [Create a basic launch template \(p. 34\)](#)
- [Specify tags that tag instances at launch \(p. 35\)](#)
- [Specify an IAM role to pass to instances \(p. 35\)](#)
- [Assign public IP addresses \(p. 35\)](#)
- [Specify a user data script that configures instances at launch \(p. 35\)](#)
- [Specify a block device mapping \(p. 36\)](#)
- [Specify Dedicated Hosts to bring software licenses from external vendors \(p. 36\)](#)
- [Specify an existing network interface \(p. 36\)](#)
- [Create multiple network interfaces \(p. 36\)](#)
- [Manage your launch templates \(p. 37\)](#)
- [Update an Auto Scaling group to use a launch template \(p. 39\)](#)

Example usage

```
{  
  "LaunchTemplateName": "my-template-for-auto-scaling",  
  "VersionDescription": "test description",  
  "LaunchTemplateData": {  
    "ImageId": "ami-04d5cc9b88example",
```

```
  "InstanceType": "t2.micro",
  "SecurityGroupIds": [
    "sg-903004f88example"
  ],
  "KeyName": "MyKeyPair",
  "Monitoring": {
    "Enabled": true
  },
  "Placement": {
    "Tenancy": "dedicated"
  },
  "CreditSpecification": {
    "CpuCredits": "unlimited"
  },
  "MetadataOptions": {
    "HttpTokens": "required",
    "HttpPutResponseHopLimit": 1,
    "HttpEndpoint": "enabled"
  }
}
```

Create a basic launch template

To create a basic launch template, use the [create-launch-template](#) command as follows, with these modifications:

- Replace `ami-04d5cc9b88example` with the ID of the AMI from which to launch the instances.
- Replace `t2.micro` with an instance type that is compatible with the AMI that you specified.

This example creates a launch template with the name `my-template-for-auto-scaling`. If the instances created by this launch template are launched in a default VPC, they receive a public IP address by default. If the instances are launched in a nondefault VPC, they do not receive a public IP address by default.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
  version-description version1 \
  --launch-template-data '{"ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}'
```

For more information about quoting JSON-formatted parameters, see [Using quotation marks with strings in the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

Alternatively, you can specify the JSON-formatted parameters in a configuration file.

The following example creates a basic launch template, referencing a configuration file for launch template parameter values.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
  version-description version1 \
  --launch-template-data file://config.json
```

Contents of config.json:

```
{ "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro" }
```

Specify tags that tag instances at launch

The following example adds a tag (for example, purpose=webserver) to instances at launch.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '[{"TagSpecifications": [{"ResourceType": "instance", "Tags":  
[{"Key": "purpose", "Value": "webserver"}]}], "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}]
```

Note

If you specify instance tags in your launch template and then you choose to propagate your Auto Scaling group's tags to its instances, all the tags are merged. If the same tag key is specified for a tag in your launch template and a tag in your Auto Scaling group, then the tag value from the group takes precedence.

Specify an IAM role to pass to instances

The following example specifies the name of the instance profile associated with the IAM role to pass to instances at launch. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '[{"IamInstanceProfile": {"Name": "my-instance-  
profile"}, "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}]
```

Assign public IP addresses

The following [create-launch-template](#) example configures the launch template to assign public addresses to instances launched in a nondefault VPC.

Note

When you specify a network interface, specify a value for Groups that corresponds to security groups for the VPC that your Auto Scaling group will launch instances into. Specify the VPC subnets as properties of the Auto Scaling group.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '[{"NetworkInterfaces":  
[{"DeviceIndex": 0, "AssociatePublicIpAddress": true, "Groups":  
["sg-903004f88example"], "DeleteOnTermination": true}], "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}]
```

Specify a user data script that configures instances at launch

The following example specifies a user data script as a base64-encoded string that configures instances at launch. The [create-launch-template](#) command requires base64-encoded user data.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data  
[{"UserData": "IyEvYmluL2Jhc..."}, "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}]
```

Specify a block device mapping

The following [create-launch-template](#) example creates a launch template with a block device mapping: a 22-gigabyte EBS volume mapped to /dev/xvdcz. The /dev/xvdcz volume uses the General Purpose SSD (gp2) volume type and is deleted when terminating the instance it is attached to.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '{"BlockDeviceMappings": [{"DeviceName": "/dev/xvdcz", "Ebs":  
{"VolumeSize": 22, "VolumeType": "gp2", "DeleteOnTermination": true}}], "ImageId": "ami-04d5cc9b88example", "In
```

Specify Dedicated Hosts to bring software licenses from external vendors

If you specify *host* tenancy, you can specify a host resource group and a License Manager license configuration to bring eligible software licenses from external vendors. Then, you can use the licenses on EC2 instances by using the following [create-launch-template](#) command.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '{"Placement":  
{"Tenancy": "host", "HostResourceGroupArn": "arn", "LicenseSpecifications":  
[{"LicenseConfigurationArn": "arn"}], "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}'
```

Specify an existing network interface

The following [create-launch-template](#) example configures the primary network interface to use an existing network interface.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '{"NetworkInterfaces": [{"DeviceIndex": 0, "NetworkInterfaceId": "eni-  
b9a5ac93", "DeleteOnTermination": false}], "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}'
```

Create multiple network interfaces

The following [create-launch-template](#) example adds a secondary network interface. The primary network interface has a device index of 0, and the secondary network interface has a device index of 1.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '{"NetworkInterfaces": [{"DeviceIndex": 0, "Groups":  
["sg-903004f88example"], "DeleteOnTermination": true}, {"DeviceIndex": 1, "Groups":  
["sg-903004f88example"], "DeleteOnTermination": true}], "ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro"}'
```

If you use an instance type that supports multiple network cards and Elastic Fabric Adapters (EFAs), you can add a secondary interface to a secondary network card and enable EFA by using the following [create-launch-template](#) command. For more information, see [Adding an EFA to a launch template](#) in the [Amazon EC2 User Guide for Linux Instances](#).

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --  
version-description version1 \  
--launch-template-data '{"NetworkInterfaces":  
[{"NetworkCardIndex": 0, "DeviceIndex": 0, "Groups":
```

```
["sg-7c2270198example"], "InterfaceType": "efa", "DeleteOnTermination": true},  
{"NetworkCardIndex": 1, "DeviceIndex": 1, "Groups":  
["sg-7c2270198example"], "InterfaceType": "efa", "DeleteOnTermination": true}], "ImageId": "ami-09d95fab7fexample
```

Warning

The p4d.24xlarge instance type incurs higher costs than the other examples in this section. For more information about pricing for P4d instances, see [Amazon EC2 P4d Instances pricing](#).

Note

Attaching multiple network interfaces from the same subnet to an instance can introduce asymmetric routing, especially on instances using a variant of non-Amazon Linux. If you need this type of configuration, you must configure the secondary network interface within the OS. For an example, see [How can I make my secondary network interface work in my Ubuntu EC2 instance?](#) in the AWS Knowledge Center.

Manage your launch templates

The AWS CLI includes several other commands that help you manage your launch templates.

Contents

- [List and describe your launch templates \(p. 37\)](#)
- [Create a launch template version \(p. 38\)](#)
- [Delete a launch template version \(p. 38\)](#)
- [Delete a launch template \(p. 39\)](#)

List and describe your launch templates

You can use two AWS CLI commands to get information about your launch templates: [describe-launch-templates](#) and [describe-launch-template-versions](#).

The [describe-launch-templates](#) command enables you to get a list of any of the launch templates that you have created. You can use an option to filter results on a launch template name, create time, tag key, or tag key-value combination. This command returns summary information about any of your launch templates, including the launch template identifier, latest version, and default version.

The following example provides a summary of the specified launch template.

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

The following is an example response.

```
{  
  "LaunchTemplates": [  
    {  
      "LaunchTemplateId": "lt-068f72b729example",  
      "LaunchTemplateName": "my-template-for-auto-scaling",  
      "CreateTime": "2020-02-28T19:52:27.000Z",  
      "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
      "DefaultVersionNumber": 1,  
      "LatestVersionNumber": 1  
    }  
  ]  
}
```

If you don't use the `--launch-template-names` option to limit the output to one launch template, information on all of your launch templates is returned.

The following [describe-launch-template-versions](#) command provides information describing the versions of the specified launch template *my-template-for-auto-scaling*.

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

The following is an example response.

```
{  
    "LaunchTemplateVersions": [  
        {  
            "VersionDescription": "version1",  
            "LaunchTemplateId": "lt-068f72b729example",  
            "LaunchTemplateName": "my-template-for-auto-scaling",  
            "VersionNumber": 1,  
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",  
            "LaunchTemplateData": {  
                "TagSpecifications": [  
                    {  
                        "ResourceType": "instance",  
                        "Tags": [  
                            {  
                                "Key": "purpose",  
                                "Value": "webserver"  
                            }  
                        ]  
                    }  
                ],  
                "ImageId": "ami-04d5cc9b88example",  
                "InstanceType": "t2.micro",  
                "NetworkInterfaces": [  
                    {  
                        "DeviceIndex": 0,  
                        "DeleteOnTermination": true,  
                        "Groups": [  
                            "sg-903004f88example"  
                        ],  
                        "AssociatePublicIpAddress": true  
                    }  
                ],  
                "DefaultVersion": true,  
                "CreateTime": "2020-02-28T19:52:27.000Z"  
            }  
        }  
    ]  
}
```

Create a launch template version

The following [create-launch-template-version](#) command creates a new launch template version based on version 1 of the launch template and specifies a different AMI ID.

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --version-description version2 \  
    --source-version 1 --launch-template-data "ImageId=ami-c998b6b2example"
```

To set the default version of the launch template, use the [modify-launch-template](#) command.

Delete a launch template version

The following [delete-launch-template-versions](#) command deletes the specified launch template version.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --  
versions 1
```

Delete a launch template

If you no longer require a launch template, you can delete it using the following [delete-launch-template](#) command. Deleting a launch template deletes all of its versions.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

Update an Auto Scaling group to use a launch template

You can use the [update-auto-scaling-group](#) command to add a launch template to an existing Auto Scaling group.

Note

If you switch your Auto Scaling group from using a launch configuration, be sure that your permissions are up to date. In order to use a launch template, you need specific [permissions](#).

Update an Auto Scaling group to use the latest version of a launch template

The following [update-auto-scaling-group](#) command updates the specified Auto Scaling group to use the latest version of the specified launch template.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

Update an Auto Scaling group to use a specific version of a launch template

The following [update-auto-scaling-group](#) command updates the specified Auto Scaling group to use a specific version of the specified launch template.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Launch configurations

Important

We strongly recommend that you do not use launch configurations. They do not provide full functionality for Amazon EC2 Auto Scaling or Amazon EC2. We provide information about launch configurations for customers who have not yet migrated from launch configurations to launch templates. For details about how long launch configurations will support new instance types and how to migrate to launch templates using the console, see [Amazon EC2 Auto Scaling will no longer add support for new EC2 features to Launch Configurations](#) on the AWS Compute Blog.

A *launch configuration* is an instance configuration template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances. Include the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping. If you've launched an EC2 instance before, you specified the same information in order to launch the instance.

You can specify your launch configuration with multiple Auto Scaling groups. However, you can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it. To change the launch configuration for an Auto Scaling group, you must create a launch configuration and then update your Auto Scaling group with it.

Keep in mind that whenever you create an Auto Scaling group, you must specify a launch configuration, a launch template, or an EC2 instance. When you create an Auto Scaling group using an EC2 instance, Amazon EC2 Auto Scaling automatically creates a launch configuration for you and associates it with the Auto Scaling group. For more information, see [Create an Auto Scaling group using parameters from an existing instance \(p. 85\)](#). Alternatively, if you are using launch templates, you can specify a launch template instead of a launch configuration or an EC2 instance. For more information, see [Launch templates \(p. 21\)](#).

Contents

- [Create a launch configuration \(p. 40\)](#)
- [Create a launch configuration using an EC2 instance \(p. 44\)](#)
- [Change the launch configuration for an Auto Scaling group \(p. 47\)](#)
- [Configure instance tenancy with a launch configuration \(p. 48\)](#)

Create a launch configuration

Important

We strongly recommend that you do not use launch configurations. They do not provide full functionality for Amazon EC2 Auto Scaling or Amazon EC2. We provide information about launch configurations for customers who have not yet migrated from launch configurations to launch templates. For details about how long launch configurations will support new instance types and how to migrate to launch templates using the console, see [Amazon EC2 Auto Scaling will no longer add support for new EC2 features to Launch Configurations](#) on the AWS Compute Blog.

When you create a launch configuration, you must specify information about the EC2 instances to launch. Include the Amazon Machine Image (AMI), instance type, key pair, security groups, and block

device mapping. Alternatively, you can create a launch configuration using attributes from a running EC2 instance. For more information, see [Create a launch configuration using an EC2 instance \(p. 44\)](#).

After you create a launch configuration, you can create an Auto Scaling group. For more information, see [Create an Auto Scaling group using a launch configuration \(p. 83\)](#).

An Auto Scaling group is associated with one launch configuration at a time, and you can't modify a launch configuration after you've created it. Therefore, if you want to change the launch configuration for an existing Auto Scaling group, you must update it with the new launch configuration. For more information, see [Change the launch configuration for an Auto Scaling group \(p. 47\)](#).

Contents

- [Create a launch configuration \(console\) \(p. 41\)](#)
- [Create a launch configuration \(AWS CLI\) \(p. 42\)](#)
- [Configure the instance metadata options \(p. 42\)](#)

Create a launch configuration (console)

To create a launch configuration (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
3. In the navigation bar, select your AWS Region.
4. Choose **Create launch configuration**, and enter a name for your launch configuration.
5. For **Amazon machine image (AMI)**, choose an AMI. To find a specific AMI, you can [find a suitable AMI](#), make note of its ID, and enter the ID as search criteria.

To get the ID of the Amazon Linux 2 AMI:

- a. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
- b. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch instances**.
- c. On the **Quick Start** tab of the **Choose an Amazon Machine Image** page, note the ID of the AMI next to **Amazon Linux 2 AMI (HVM)**.
6. For **Instance type**, select a hardware configuration for your instances.
7. Under **Additional configuration**, pay attention to the following fields:
 - a. (Optional) For **Purchasing option**, you can choose **Request Spot Instances** to request Spot Instances at the Spot price, capped at the On-Demand price. Optionally, you can specify a maximum price per instance hour for your Spot Instances.

Note

Spot Instances are a cost-effective choice compared to On-Demand Instances, if you can be flexible about when your applications run and if your applications can be interrupted. For more information, see [Request Spot Instances for fault-tolerant and flexible applications \(p. 32\)](#).

- b. (Optional) For **IAM instance profile**, choose a role to associate with the instances. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).
- c. (Optional) For **Monitoring**, choose whether to enable the instances to publish metric data at 1-minute intervals to Amazon CloudWatch by enabling detailed monitoring. Additional charges apply. For more information, see [Configure monitoring for Auto Scaling instances \(p. 265\)](#).
- d. (Optional) For **Advanced details, User data**, you can specify user data to configure an instance during launch, or to run a configuration script after the instance starts.

- e. (Optional) For **Advanced details**, **IP address type**, choose whether to assign a [public IP address](#) to the group's instances. If you do not set a value, the default is to use the auto-assign public IP settings of the subnets that your instances are launched into.
8. (Optional) For **Storage (volumes)**, if you don't need additional storage, you can skip this section. Otherwise, to specify volumes to attach to the instances in addition to the volumes specified by the AMI, choose **Add new volume**. Then choose the desired options and associated values for **Devices**, **Snapshot**, **Size**, **Volume type**, **IOPS**, **Throughput**, **Delete on termination**, and **Encrypted**.
9. For **Security groups**, create or select the security group to associate with the group's instances. If you leave the **Create a new security group** option selected, a default SSH rule is configured for Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.
10. For **Key pair (login)**, choose an option under **Key pair options**.

If you've already configured an Amazon EC2 instance key pair, you can choose it here.

If you don't already have an Amazon EC2 instance key pair, choose **Create a new key pair** and give it a recognizable name. Choose **Download key pair** to download the key pair to your computer.

Important

If you need to connect to your instances, do not choose **Proceed without a key pair**.

11. Select the acknowledgment check box, and then choose **Create launch configuration**.

Create a launch configuration (AWS CLI)

To create a launch configuration using the command line

You can use one of the following commands:

- [create-launch-configuration \(AWS CLI\)](#)
- [New-ASLaunchConfiguration \(AWS Tools for Windows PowerShell\)](#)

Configure the instance metadata options

Amazon EC2 Auto Scaling supports configuring the Instance Metadata Service (IMDS) in launch configurations. This gives you the option of using launch configurations to configure the Amazon EC2 instances in your Auto Scaling groups to require Instance Metadata Service Version 2 (IMDSv2), which is a session-oriented method for requesting instance metadata. For details about IMDSv2's advantages, see this article on the AWS Blog about [enhancements to add defense in depth to the EC2 instance metadata service](#).

You can configure IMDS to support both IMDSv2 and IMDSv1 (the default), or to require the use of IMDSv2. If you are using the AWS CLI or one of the SDKs to configure IMDS, you must use the latest version of the AWS CLI or the SDK to require the use of IMDSv2.

You can configure your launch configuration for the following:

- Require the use of IMDSv2 when requesting instance metadata
- Specify the PUT response hop limit
- Turn off access to instance metadata

You can find more details on configuring the Instance Metadata Service in the following topic: [Configuring the instance metadata service](#) in the *Amazon EC2 User Guide for Linux Instances*.

Use the following procedure to configure IMDS options in a launch configuration. After you create your launch configuration, you can associate it with your Auto Scaling group. If you associate the launch configuration with an existing Auto Scaling group, the existing launch configuration is disassociated from the Auto Scaling group, and existing instances will require replacement to use the IMDS options that you specified in the new launch configuration. For more information, see [Change the launch configuration for an Auto Scaling group \(p. 47\)](#).

To configure IMDS in a launch configuration (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
3. In the navigation bar, select your AWS Region.
4. Choose **Create launch configuration**, and create the launch configuration the usual way. Include the ID of the Amazon Machine Image (AMI), the instance type, and optionally, a key pair, one or more security groups, and any additional EBS volumes or instance store volumes for your instances.
5. To configure instance metadata options for all of the instances associated with this launch configuration, in **Additional configuration**, under **Advanced details**, do the following:
 - a. For **Metadata accessible**, choose whether to enable or disable access to the HTTP endpoint of the instance metadata service. By default, the HTTP endpoint is enabled. If you choose to disable the endpoint, access to your instance metadata is turned off. You can specify the condition to require IMDSv2 only when the HTTP endpoint is enabled.
 - b. For **Metadata version**, you can choose to require the use of Instance Metadata Service Version 2 (IMDSv2) when requesting instance metadata. If you do not specify a value, the default is to support both IMDSv1 and IMDSv2.
 - c. For **Metadata token response hop limit**, you can set the allowable number of network hops for the metadata token. If you do not specify a value, the default is 1.
6. When you have finished, choose **Create launch configuration**.

To require the use of IMDSv2 in a launch configuration using the AWS CLI

Use the following `create-launch-configuration` command with `--metadata-options` set to `HttpTokens=required`. When you specify a value for `HttpTokens`, you must also set `HttpEndpoint` to `enabled`. Because the secure token header is set to `required` for metadata retrieval requests, this option in the instance to require using IMDSv2 when requesting instance metadata.

```
aws autoscaling create-launch-configuration \
--launch-configuration-name my-lc-with-imds2 \
--image-id ami-01e24be29428c15b2 \
--instance-type t2.micro \
...
--metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

To turn off access to instance metadata

Use the following `create-launch-configuration` command to turn off access to instance metadata. You can turn access back on later by using the `modify-instance-metadata-options` command.

```
aws autoscaling create-launch-configuration \
--launch-configuration-name my-lc-with-imds-disabled \
--image-id ami-01e24be29428c15b2 \
--instance-type t2.micro \
...
--metadata-options "HttpEndpoint=disabled"
```

Create a launch configuration using an EC2 instance

Amazon EC2 Auto Scaling provides you with an option to create a launch configuration using the attributes from a running EC2 instance.

If the specified instance has properties that are not currently supported by launch configurations, the instances launched by the Auto Scaling group might not be identical to the original EC2 instance.

There are differences between creating a launch configuration from scratch and creating a launch configuration from an existing EC2 instance. When you create a launch configuration from scratch, you specify the image ID, instance type, optional resources (such as storage devices), and optional settings (like monitoring). When you create a launch configuration from a running instance, Amazon EC2 Auto Scaling derives attributes for the launch configuration from the specified instance. Attributes are also derived from the block device mapping for the AMI from which the instance was launched, ignoring any additional block devices that were added after launch.

When you create a launch configuration using a running instance, you can override the following attributes by specifying them as part of the same request: AMI, block devices, key pair, instance profile, instance type, kernel, instance monitoring, placement tenancy, ramdisk, security groups, Spot (max) price, user data, whether the instance has a public IP address, and whether the instance is EBS-optimized.

Tip

You can [create an Auto Scaling group directly from an EC2 instance \(p. 85\)](#). When you use this feature, Amazon EC2 Auto Scaling automatically creates a launch configuration for you as well.

The following examples show you to create a launch configuration from an EC2 instance.

Examples

- [Create a launch configuration using an EC2 instance \(p. 44\)](#)
- [Create a launch configuration from an instance and override the block devices \(AWS CLI\) \(p. 45\)](#)
- [Create a launch configuration and override the instance type \(AWS CLI\) \(p. 46\)](#)

Create a launch configuration using an EC2 instance

To create a launch configuration using the attributes of an existing EC2 instance, specify the ID of the instance.

Important

The AMI used to launch the specified instance must still exist.

Create a launch configuration from an EC2 instance (console)

You can use the console to create a launch configuration and an Auto Scaling group from a running EC2 instance and add the instance to the new Auto Scaling group. For more information, see [Attach EC2 instances to your Auto Scaling group \(p. 126\)](#).

Create a launch configuration from an EC2 instance (AWS CLI)

Use the following `create-launch-configuration` command to create a launch configuration from an instance using the same attributes as the instance. Any block devices added after launch are ignored.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance  
--instance-id i-a8e09d9c
```

You can use the following [describe-launch-configurations](#) command to describe the launch configuration and verify that its attributes match those of the instance.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance
```

The following is an example response.

```
{  
  "LaunchConfigurations": [  
    {  
      "UserData": null,  
      "EbsOptimized": false,  
      "LaunchConfigurationARN": "arn",  
      "InstanceMonitoring": {  
        "Enabled": false  
      },  
      "ImageId": "ami-05355a6c",  
      "CreatedTime": "2014-12-29T16:14:50.382Z",  
      "BlockDeviceMappings": [],  
      "KeyName": "my-key-pair",  
      "SecurityGroups": [  
        "sg-8422d1eb"  
      ],  
      "LaunchConfigurationName": "my-lc-from-instance",  
      "KernelId": "null",  
      "RamdiskId": null,  
      "InstanceType": "t1.micro",  
      "AssociatePublicIpAddress": true  
    }  
  ]  
}
```

Create a launch configuration from an instance and override the block devices (AWS CLI)

By default, Amazon EC2 Auto Scaling uses the attributes from the EC2 instance that you specify to create the launch configuration. However, the block devices come from the AMI used to launch the instance, not the instance. To add block devices to the launch configuration, override the block device mapping for the launch configuration.

Important

The AMI used to launch the specified instance must still exist.

Create a launch configuration and override the block devices

Use the following [create-launch-configuration](#) command to create a launch configuration using an EC2 instance but with a custom block device mapping.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-bdm --instance-id i-a8e09d9c \  
  --block-device-mappings "[{\\"DeviceName\\":\"/dev/sda1\", \"Ebs\":{\"SnapshotId\\":\"snap-3decf207\"}, {\\"DeviceName\\\":\"/dev/sdf\", \"Ebs\":{\"SnapshotId\\\":\"snap-eed6ac86\"}}}]"
```

Use the following [describe-launch-configurations](#) command to describe the launch configuration and verify that it uses your custom block device mapping.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-bdm
```

The following example response describes the launch configuration.

```
{  
    "LaunchConfigurations": [  
        {  
            "UserData": null,  
            "EbsOptimized": false,  
            "LaunchConfigurationARN": "arn:",  
            "InstanceMonitoring": {  
                "Enabled": false  
            },  
            "ImageId": "ami-c49c0dac",  
            "CreatedTime": "2015-01-07T14:51:26.065Z",  
            "BlockDeviceMappings": [  
                {  
                    "DeviceName": "/dev/sda1",  
                    "Ebs": {  
                        "SnapshotId": "snap-3decf207"  
                    }  
                },  
                {  
                    "DeviceName": "/dev/sdf",  
                    "Ebs": {  
                        "SnapshotId": "snap-eed6ac86"  
                    }  
                }  
            ],  
            "KeyName": "my-key-pair",  
            "SecurityGroups": [  
                "sg-8637d3e3"  
            ],  
            "LaunchConfigurationName": "my-lc-from-instance-bdm",  
            "KernelId": null,  
            "RamdiskId": null,  
            "InstanceType": "t1.micro",  
            "AssociatePublicIpAddress": true  
        }  
    ]  
}
```

Create a launch configuration and override the instance type (AWS CLI)

By default, Amazon EC2 Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration. Depending on your requirements, you might want to override attributes from the instance and use the values that you need. For example, you can override the instance type.

Important

The AMI used to launch the specified instance must still exist.

Create a launch configuration and override the instance type

Use the following [create-launch-configuration](#) command to create a launch configuration using an EC2 instance but with a different instance type (for example `t2.medium`) than the instance (for example `t2.micro`).

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-changetype \
--instance-id i-a8e09d9c --instance-type t2.medium
```

Use the following [describe-launch-configurations](#) command to describe the launch configuration and verify that the instance type was overridden.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-changetype
```

The following example response describes the launch configuration.

```
{  
    "LaunchConfigurations": [  
        {  
            "UserData": null,  
            "EbsOptimized": false,  
            "LaunchConfigurationARN": "arn:",  
            "InstanceMonitoring": {  
                "Enabled": false  
            },  
            "ImageId": "ami-05355a6c",  
            "CreatedTime": "2014-12-29T16:14:50.382Z",  
            "BlockDeviceMappings": [],  
            "KeyName": "my-key-pair",  
            "SecurityGroups": [  
                "sg-8422d1eb"  
            ],  
            "LaunchConfigurationName": "my-lc-from-instance-changetype",  
            "KernelId": "null",  
            "RamdiskId": null,  
            "InstanceType": "t2.medium",  
            "AssociatePublicIpAddress": true  
        }  
    ]  
}
```

Change the launch configuration for an Auto Scaling group

An Auto Scaling group is associated with one launch configuration at a time, and you can't modify a launch configuration after you've created it. To change the launch configuration for an Auto Scaling group, use an existing launch configuration as the basis for a new launch configuration. Then, update the Auto Scaling group to use the new launch configuration.

After you change the launch configuration for an Auto Scaling group, any new instances are launched using the new configuration options, but existing instances are not affected. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow auto scaling to gradually replace older instances with newer instances based on your [termination policies \(p. 230\)](#).

Note

You can also replace all instances in the Auto Scaling group to launch new instances that use the new launch configuration. For more information, see [Replace Auto Scaling instances \(p. 99\)](#).

To change the launch configuration for an Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.

2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
3. Select the launch configuration and choose **Actions, Copy launch configuration**. This sets up a new launch configuration with the same options as the original, but with "Copy" added to the name.
4. On the **Copy Launch Configuration** page, edit the configuration options as needed and choose **Create launch configuration**.
5. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
6. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom part of the page, showing information about the group that's selected.

7. On the **Details** tab, choose **Launch configuration, Edit**.
8. For **Launch configuration**, select the new launch configuration.
9. When you have finished, choose **Update**.

To change the launch configuration for an Auto Scaling group (AWS CLI)

1. Describe the current launch configuration using the [describe-launch-configurations](#) command.
2. Create a new launch configuration using the [create-launch-configuration](#) command.
3. Update the launch configuration for the Auto Scaling group using the [update-auto-scaling-group](#) command with the `--launch-configuration-names` parameter.

To change the launch configuration for an Auto Scaling group (Tools for Windows PowerShell)

1. Describe the current launch configuration using the [Get-ASLaunchConfiguration](#) command.
2. Create a new launch configuration using the [New-ASLaunchConfiguration](#) command.
3. Update the launch configuration for the Auto Scaling group using the [Update-ASAutoScalingGroup](#) command with the `-LaunchConfigurationName` parameter.

Configure instance tenancy with a launch configuration

Tenancy defines how EC2 instances are distributed across physical hardware and affects pricing. There are three tenancy options available:

- Shared (default) — Multiple AWS accounts may share the same physical hardware.
- Dedicated Instance (dedicated) — Your instance runs on single-tenant hardware.
- Dedicated Host (host) — Your instance runs on a physical server with EC2 instance capacity fully dedicated to your use, an isolated server with configurations that you can control.

This topic describes how to launch Dedicated Instances in your Auto Scaling group by specifying settings in a launch configuration. For pricing information and to learn more about Dedicated Instances, see the [Amazon EC2 dedicated instances](#) product page and [Dedicated Instances](#) in the [Amazon EC2 User Guide for Linux Instances](#).

You can configure tenancy for EC2 instances using a launch configuration or launch template. However, the host tenancy value cannot be used with a launch configuration. Use the default or dedicated tenancy values only.

Important

To use a tenancy value of host, you must use a launch template. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#). Before launching Dedicated Hosts, we recommend that you become familiar with launching and managing Dedicated Hosts using [AWS License Manager](#). For more information, see the [License Manager User Guide](#).

When you create a VPC, by default its tenancy attribute is set to default. In such a VPC, you can launch instances with a tenancy value of dedicated so that they run as single-tenancy instances. Otherwise, they run as shared-tenancy instances by default. If you set the tenancy attribute of a VPC to dedicated, all instances launched in the VPC run as single-tenancy instances.

When you create a launch configuration, the default value for the instance placement tenancy is null and the instance tenancy is controlled by the tenancy attribute of the VPC. You can specify the instance placement tenancy for your launch configuration as default or dedicated using the [create-launch-configuration](#) CLI command with the --placement-tenancy option.

The following table summarizes the instance placement tenancy of the Auto Scaling instances launched in a VPC.

Launch configuration tenancy	VPC tenancy = default	VPC tenancy = dedicated
not specified	shared-tenancy instances	Dedicated Instances
default	shared-tenancy instances	Dedicated Instances
dedicated	Dedicated Instances	Dedicated Instances

To create a launch configuration that creates Dedicated Instances (AWS CLI)

Use the following [create-launch-configuration](#) command to create a launch configuration that sets the launch configuration tenancy to dedicated.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-launch-config --placement-tenancy dedicated --image-id ...
```

You can use the following [describe-launch-configurations](#) command to verify the instance placement tenancy of the launch configuration.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-launch-config
```

The following is example output for a launch configuration that creates Dedicated Instances. The PlacementTenancy parameter is only part of the output for this command when you explicitly set the instance placement tenancy.

```
{
  "LaunchConfigurations": [
    {
      "UserData": null,
      "EbsOptimized": false,
      "PlacementTenancy": "dedicated",
      "LaunchConfigurationARN": "arn",
      "InstanceMonitoring": {
        "Enabled": true
      },
      "ImageId": "ami-b5a7ea85",
      "CreatedTime": "2020-03-08T23:39:49.011Z",
    }
  ]
}
```

```
    "BlockDeviceMappings": [],
    "KeyName": null,
    "SecurityGroups": [],
    "LaunchConfigurationName": "my-launch-config",
    "KernelId": null,
    "RamdiskId": null,
    "InstanceType": "m3.medium"
  ]
}
```

Auto Scaling groups

Note

If you are new to Auto Scaling groups, you can learn more in the [Get started with Amazon EC2 Auto Scaling \(p. 14\)](#) tutorial. You start by creating a launch template or a launch configuration and then use it to create an Auto Scaling group in which all instances have the same instance attributes. For example, you can set the following instance attributes by specifying them as part of the launch template or launch configuration: Amazon Machine Image (AMI), instance type, block storage devices, SSH key pair, and security groups that control the inbound and outbound traffic for an instance.

An *Auto Scaling group* contains a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also lets you use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity. You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its desired capacity. It maintains this number of instances by performing periodic health checks on the instances in the group. The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).

You can use scaling policies to increase or decrease the number of instances in your group dynamically to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group, between the minimum and maximum capacity values that you specify, and launches or terminates the instances as needed. You can also scale on a schedule. For more information, see [Scale the size of your Auto Scaling group \(p. 121\)](#).

An Auto Scaling group can launch On-Demand Instances, Spot Instances, or both. You can specify multiple purchase options for your Auto Scaling group only when you use a launch template. We recommend using launch templates instead of launch configurations to make sure that you can use the latest features of Amazon EC2 Auto Scaling and Amazon EC2.

Spot Instances provide you with access to unused EC2 capacity at steep discounts relative to On-Demand prices. For more information, see [Amazon EC2 Spot Instances](#). There are key differences between Spot Instances and On-Demand Instances:

- The price for Spot Instances varies based on demand
- Amazon EC2 can terminate an individual Spot Instance as the availability of, or price for, Spot Instances changes

When a Spot Instance is terminated, the Auto Scaling group attempts to launch a replacement instance to maintain the desired capacity for the group.

When instances are launched, if you specified multiple Availability Zones, the desired capacity is distributed across these Availability Zones. If a scaling action occurs, Amazon EC2 Auto Scaling automatically maintains balance across all of the Availability Zones that you specify.

Contents

- [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#)
- [Create an Auto Scaling group using attribute-based instance type selection \(p. 75\)](#)

- [Create an Auto Scaling group using a launch template \(p. 81\)](#)
- [Create an Auto Scaling group using a launch configuration \(p. 83\)](#)
- [Create an Auto Scaling group using parameters from an existing instance \(p. 85\)](#)
- [Create an Auto Scaling group using the Amazon EC2 launch wizard \(p. 89\)](#)
- [Tag Auto Scaling groups and instances \(p. 91\)](#)
- [Replace Auto Scaling instances \(p. 99\)](#)
- [Delete your Auto Scaling infrastructure \(p. 117\)](#)

Auto Scaling groups with multiple instance types and purchase options

You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for EC2 instances, while making sure that you obtain the desired scale and performance for your application.

You first specify the common configuration parameters in a launch template, and choose it when you create an Auto Scaling group. When you configure the Auto Scaling group, you can:

- Choose one or more instance types for the group (optionally overriding the instance type that is specified by the launch template).
- Define multiple launch templates to allow instances with different CPU architectures (for example, Arm and x86) to launch in the same Auto Scaling group.
- Assign each instance type an individual weight. Doing so might be useful, for example, if the instance types offer different vCPU, memory, storage, or network bandwidth capabilities.
- Prioritize instance types that can benefit from Savings Plan or Reserved Instance discount pricing.
- Specify how much On-Demand and Spot capacity to launch, and specify an optional On-Demand base portion.
- Define how Amazon EC2 Auto Scaling should distribute your On-Demand and Spot capacity across instance types.
- Enable Capacity Rebalancing. When you turn on Capacity Rebalancing, Amazon EC2 Auto Scaling attempts to launch a Spot Instance whenever the Amazon EC2 Spot service notifies that a Spot Instance is at an elevated risk of interruption. After launching a new instance, it then terminates an old instance. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).

You enhance availability by deploying your application across multiple instance types running in multiple Availability Zones. You can use just one instance type, but it is a best practice to use a few instance types to allow Amazon EC2 Auto Scaling to launch another instance type in the event that there is insufficient instance capacity in your chosen Availability Zones. With Spot Instances, if there is insufficient instance capacity, Amazon EC2 Auto Scaling keeps trying in other Spot Instance pools (determined by your choice of instance types and allocation strategy) rather than launching On-Demand Instances, so that you can leverage the cost savings of Spot Instances.

There are two ways to associate multiple instance types with your Auto Scaling group configuration:

- Manually add instance types as described in this topic.
- Choose a set of instance attributes to use as criteria for selecting the instance types that your Auto Scaling group uses. This is known as *attribute-based instance type selection*. For more information, see [Using attribute-based instance type selection \(p. 75\)](#).

Contents

- [Allocation strategies \(p. 53\)](#)
- [Control the proportion of On-Demand Instances \(p. 55\)](#)
- [Best practices for Spot Instances \(p. 56\)](#)
- [Prerequisites \(p. 57\)](#)
- [Create an Auto Scaling group with Spot and On-Demand Instances \(console\) \(p. 57\)](#)
- [Configure Spot allocation strategies \(AWS CLI\) \(p. 59\)](#)
- [Verify whether your Auto Scaling group is configured correctly and that the group has launched instances \(AWS CLI\) \(p. 64\)](#)
- [Configure overrides \(p. 64\)](#)

Allocation strategies

The following allocation strategies determine how the Auto Scaling group fulfills your On-Demand and Spot capacities from the possible instance types.

Amazon EC2 Auto Scaling first tries to ensure that your instances are evenly balanced across the Availability Zones that you specified. Then, it launches instance types according to the allocation strategy that is specified.

Spot Instances

Amazon EC2 Auto Scaling provides the following allocation strategies that can be used for Spot Instances:

capacity-optimized

Amazon EC2 Auto Scaling allocates your instances from the Spot Instance pool with optimal capacity for the number of instances that are launching. Deploying in this way helps you make the most efficient use of spare EC2 capacity.

With Spot Instances, the pricing changes slowly over time based on long-term trends in supply and demand, but capacity fluctuates in real time. The capacity-optimized strategy automatically launches Spot Instances into the most available pools by looking at real-time capacity data and predicting which are the most available. This works well for workloads such as big data and analytics, image and media rendering, and machine learning. It also works well for high performance computing that may have a higher cost of interruption associated with restarting work and checkpointing. By offering the possibility of fewer interruptions, the capacity-optimized strategy can lower the overall cost of your workload.

Alternatively, you can use the capacity-optimized-prioritized allocation strategy and then set the order of instance types in the list of launch template overrides from highest to lowest priority (first to last in the list). Amazon EC2 Auto Scaling honors the instance type priorities on a best-effort basis but optimizes for capacity first. This is a good option for workloads where the possibility of disruption must be minimized, but also the preference for certain instance types matters.

lowest-price

Amazon EC2 Auto Scaling allocates your Spot Instances from the N number of pools per Availability Zone that you specify and from the Spot Instance pools with the lowest price in each Availability Zone.

For example, if you specify four instance types and four Availability Zones, your Auto Scaling group has access to as many as 16 Spot pools (four in each Availability Zone). If you specify two Spot pools

(N=2) for the allocation strategy, your Auto Scaling group can draw on the two cheapest pools per Availability Zone to fulfill your Spot capacity.

Note that Amazon EC2 Auto Scaling attempts to draw Spot Instances from the N number of pools that you specify on a best effort basis. If a pool runs out of Spot capacity before fulfilling your desired capacity, Amazon EC2 Auto Scaling will continue to fulfill your request by drawing from the next cheapest pool. To ensure that your desired capacity is met, you might receive Spot Instances from more than the N number of pools that you specified. Similarly, if most of the pools have no Spot capacity, you might receive your full desired capacity from fewer than the N number of pools that you specified.

To get started, we recommend choosing the capacity-optimized allocation strategy and specifying a few instance types that are appropriate for your application. In addition, you can define a range of Availability Zones for Amazon EC2 Auto Scaling to choose from when launching instances.

Optionally, you can specify a maximum price for your Spot Instances. If you don't specify a maximum price, the default maximum price is the On-Demand price, but you still receive the steep discounts provided by Spot Instances. These discounts are possible because of the stable Spot pricing that is made available using the new [Spot pricing model](#).

For more information about the allocation strategies for Spot Instances, see [Introducing the capacity-optimized allocation strategy for Amazon EC2 Spot Instances](#) in the AWS blog.

On-Demand Instances

Amazon EC2 Auto Scaling provides the following allocation strategies that can be used for On-Demand Instances:

`lowest-price`

Amazon EC2 Auto Scaling automatically deploys the cheapest instance type in each Availability Zone based on the current On-Demand price.

To ensure that your desired capacity is met, you might receive On-Demand Instances of more than one instance type in each Availability Zone, depending on how much capacity you request.

Note

Currently, Amazon EC2 Auto Scaling does not honor the `lowest-price` allocation strategy for On-Demand Instances when implementing your [termination policies \(p. 233\)](#) during scale-in events.

`prioritized`

Amazon EC2 Auto Scaling uses the order of instance types in the list of launch template overrides to determine which instance type to use first when fulfilling On-Demand capacity. For example, let's say that you specified three launch template overrides in the following order: `c5.large`, `c4.large`, and `c3.large`. When your On-Demand Instances are launched, the Auto Scaling group fulfills On-Demand capacity by starting with `c5.large`, then `c4.large`, and then `c3.large`.

Consider the following when managing the priority order of your On-Demand Instances:

You can pay for usage upfront to get significant discounts for On-Demand Instances by using either Savings Plans or Reserved Instances. For more information about Savings Plans or Reserved Instances, see the [Amazon EC2 pricing](#) page.

- With Reserved Instances, your discounted rate of the regular On-Demand Instance pricing applies if Amazon EC2 Auto Scaling launches matching instance types. That means that if you have unused Reserved Instances for `c4.large`, you can set the instance type priority to give the

highest priority for your Reserved Instances to a `c4.large` instance type. When a `c4.large` instance launches, you receive the Reserved Instance pricing.

- With Savings Plans, your discounted rate of the regular On-Demand Instance pricing applies when using either Amazon EC2 Instance Savings Plans or Compute Savings Plans. Because of the flexible nature of Savings Plans, you have greater flexibility in prioritizing your instance types. As long as you use instance types that are covered by your Savings Plan, you can set them in any order of priority and even occasionally change their order entirely, and continue to receive the discounted rate provided by your Savings Plan. To learn more about Savings Plans, see the [Savings Plans User Guide](#).

Control the proportion of On-Demand Instances

You have full control over the proportion of instances in the Auto Scaling group that are launched as On-Demand Instances. To ensure that you always have instance capacity, you can designate a percentage of the group to launch as On-Demand Instances and, optionally, a base number of On-Demand Instances to start with. If you choose to specify a base capacity of On-Demand Instances, Amazon EC2 Auto Scaling launches Spot Instances only after launching this base capacity of On-Demand Instances when the group scales out. Anything beyond the base capacity uses the On-Demand percentage to determine how many On-Demand Instances and Spot Instances to launch. You can specify any number from 0 to 100 for the On-Demand percentage.

Amazon EC2 Auto Scaling converts the percentage to the equivalent number of instances. If the result creates a fractional number, Amazon EC2 Auto Scaling rounds up to the next integer in favor of On-Demand Instances.

The behavior of the Auto Scaling group as it increases in size is as follows:

Example: Scaling behavior

Instances distribution	Total number of running instances across purchase options			
	10	20	30	40
Example 1				
On-Demand base: 10	10	10	10	10
On-Demand percentage above base: 50%	0	5	10	15
Spot percentage: 50%	0	5	10	15
Example 2				
On-Demand base: 0	0	0	0	0
On-Demand percentage above base: 0%	0	0	0	0
Spot percentage: 100%	10	20	30	40
Example 3				

Instances distribution	Total number of running instances across purchase options			
On-Demand base: 0	0	0	0	0
On-Demand percentage above base: 60%	6	12	18	24
Spot percentage: 40%	4	8	12	16
Example 4				
On-Demand base: 0	0	0	0	0
On-Demand percentage above base: 100%	10	20	30	40
Spot percentage: 0%	0	0	0	0
Example 5				
On-Demand base: 12	10	12	12	12
On-Demand percentage above base: 0%	0	0	0	0
Spot percentage: 100%	0	8	18	28

Best practices for Spot Instances

Before you create your Auto Scaling group to request Spot Instances, review [Best practices for EC2 Spot](#) in the *Amazon EC2 User Guide for Linux Instances*. Use these best practices when you plan your request so that you can provision the type of instances that you want at the lowest possible price. We also recommend that you do the following:

- Use the default maximum price, which is the On-Demand price. You pay only the Spot price for the Spot Instances that you launch. If the Spot price is within your maximum price, whether your request is fulfilled depends on availability. For more information, see [Pricing and savings](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Create your Auto Scaling group with multiple instance types. Because capacity fluctuates independently for each instance type in an Availability Zone, you can often get more compute capacity when you have instance type flexibility.
- Similarly, don't limit yourself to only the most popular instance types. Because prices adjust based on long-term demand, popular instance types (such as recently launched instance families), tend to have more price adjustments. Picking older-generation instance types that are less popular tends to result in lower costs and fewer interruptions.
- We recommend that you use the capacity-optimized or capacity-optimized-prioritized allocation strategy. This means that Amazon EC2 Auto Scaling launches instances using Spot pools

that are optimally chosen based on the available Spot capacity, which helps you reduce the possibility of a Spot interruption.

- If you choose the lowest-price allocation strategy and you run a web service, specify a high number of Spot pools, for example, N=10. Specifying a high number of Spot pools reduces the impact of Spot Instance interruptions if a pool in one of the Availability Zones becomes temporarily unavailable. If you run batch processing or other non-mission critical applications, you can specify a lower number of Spot pools, for example, N=2. This helps to ensure that you provision Spot Instances from only the very lowest priced Spot pools available per Availability Zone.

If you intend to specify a maximum price, use the AWS CLI or an SDK to create the Auto Scaling group, but be cautious. If your maximum price is lower than the Spot price for the instance types that you selected, your Spot Instances are not launched.

Prerequisites

Create a launch template. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

Verify that you have the permissions required to use a launch template. Your `ec2:RunInstances` permissions are checked when use a launch template. Your `iam:PassRole` permissions are also checked if the launch template specifies an IAM role. For more information, see [Launch template support \(p. 347\)](#).

Create an Auto Scaling group with Spot and On-Demand Instances (console)

Complete the following steps to create a fleet of Spot Instances and On-Demand Instances that you can scale.

Before you begin, confirm that you have created a launch template, as described in [Prerequisites \(p. 57\)](#).

Verify that the launch template doesn't already request Spot Instances.

To create an Auto Scaling group with Spot and On-Demand Instances

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. On the navigation bar at the top of the screen, choose the same AWS Region that you used when you created the launch template.
3. Choose **Create an Auto Scaling group**.
4. On the **Choose launch template or configuration** page, do the following:
 - a. For **Auto Scaling group name**, enter a name for your Auto Scaling group.
 - b. For **Launch template**, choose an existing launch template.
 - c. For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.
 - d. Verify that your launch template supports all of the options that you are planning to use, and then choose **Next**.
5. On the **Choose instance launch options** page, under **Network**, for **VPC**, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch template.

6. For **Availability Zones and subnets**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).
7. For **Instance type requirements**, choose **Override launch template, Manually add instance types**.
 - a. For **Instance types**, choose which types of instances can be launched. You can use our recommendations as a starting point.
 - b. (Optional) To change the order of the instance types, use the arrows. The order in which you set the instance types sets their launch priority if you choose a priority-based allocation strategy.
 - c. (Optional) To use [instance weighting \(p. 68\)](#), assign each instance type a relative weight that corresponds to how much the instance should count toward the capacity of the Auto Scaling group.
8. Under **Instance purchase options**, make updates to the purchase options as needed to lower the cost of your application by using Spot Instances.
 - a. For **Instances distribution**, specify the percentages of On-Demand Instances to Spot Instances to launch for the Auto Scaling group. If your application is stateless, fault tolerant and can handle an instance being interrupted, you can specify a higher percentage of Spot Instances.
 - b. Depending on whether you chose to launch Spot Instances, you can select the check box next to **Include On-Demand base capacity** and then specify the minimum amount of the Auto Scaling group's initial capacity that must be fulfilled by On-Demand Instances. Anything beyond the base capacity uses the **Instances distribution** settings to determine how many On-Demand Instances and Spot Instances to launch.
9. Under **Allocation strategies**, for **On-Demand allocation strategy**, choose an allocation strategy.
10. For **Spot allocation strategy**, choose an allocation strategy. We recommend that you keep the default setting of **Capacity optimized**. If you prefer not to keep the default, choose **Lowest price**, and then enter the number of lowest priced Spot Instance pools to diversify across.
11. For **Capacity rebalance**, choose whether to enable or disable Capacity Rebalancing. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).
12. Choose **Next**.

Or, you can accept the rest of the defaults, and choose **Skip to review**.

13. On the **Configure advanced options** page, configure the options as desired, and then choose **Next**.
14. (Optional) On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**.
 - a. For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see [Set capacity limits on your Auto Scaling group \(p. 122\)](#).
 - b. To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see [Target tracking scaling policies for Amazon EC2 Auto Scaling \(p. 135\)](#).
 - c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see [Use instance scale-in protection \(p. 240\)](#).
15. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).
16. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see [Tag Auto Scaling groups and instances \(p. 91\)](#).
17. On the **Review** page, choose **Create Auto Scaling group**.

Configure Spot allocation strategies (AWS CLI)

The following example configurations show how to launch Spot Instances using the different Spot allocation strategies.

Note

These examples show how to use a configuration file formatted in JSON or YAML. If you use AWS CLI version 1, you must specify a JSON-formatted configuration file. If you use AWS CLI version 2, you can specify a configuration file formatted in either YAML or JSON.

Examples

- [Example 1: Launch Spot Instances using the capacity-optimized allocation strategy \(p. 59\)](#)
- [Example 2: Launch Spot Instances using the capacity-optimized-prioritized allocation strategy \(p. 61\)](#)
- [Example 3: Launch Spot Instances using the lowest-price allocation strategy diversified over two pools \(p. 62\)](#)

Example 1: Launch Spot Instances using the capacity-optimized allocation strategy

The following `create-auto-scaling-group` command creates an Auto Scaling group that specifies the following:

- The percentage of the group to launch as On-Demand Instances (0) and a base number of On-Demand Instances to start with (1)
- The instance types to launch in priority order (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large)
- The subnets in which to launch the instances (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782), each corresponding to a different Availability Zone
- The launch template (`my-launch-template`) and the launch template version (`$Default`)

When Amazon EC2 Auto Scaling attempts to fulfill your On-Demand capacity, it launches the c5.large instance type first. The Spot Instances come from the optimal Spot pool in each Availability Zone based on Spot Instance capacity.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "MixedInstancesPolicy": {  
        "LaunchTemplate": {  
            "LaunchTemplateSpecification": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "$Default"  
            },  
            "Overrides": [  
                {  
                    "InstanceType": "c5.large"  
                }  
            ]  
        }  
    }  
}
```

```

        "InstanceType": "c5a.large"
    },
    {
        "InstanceType": "m5.large"
    },
    {
        "InstanceType": "m5a.large"
    },
    {
        "InstanceType": "c4.large"
    },
    {
        "InstanceType": "m4.large"
    },
    {
        "InstanceType": "c3.large"
    },
    {
        "InstanceType": "m3.large"
    }
]
},
"InstancesDistribution": {
    "OnDemandBaseCapacity": 1,
    "OnDemandPercentageAboveBaseCapacity": 0,
    "SpotAllocationStrategy": "capacity-optimized"
}
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

Alternatively, you can use the following [create-auto-scaling-group](#) command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

The following is an example config.yaml file.

```

---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
    LaunchTemplate:
        LaunchTemplateSpecification:
            LaunchTemplateName: my-launch-template
            Version: $Default
    Overrides:
        - InstanceType: c5.large
        - InstanceType: c5a.large
        - InstanceType: m5.large
        - InstanceType: m5a.large
        - InstanceType: c4.large
        - InstanceType: m4.large
        - InstanceType: c3.large
        - InstanceType: m3.large
    InstancesDistribution:
        OnDemandBaseCapacity: 1
        OnDemandPercentageAboveBaseCapacity: 0
        SpotAllocationStrategy: capacity-optimized

```

```
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Example 2: Launch Spot Instances using the capacity-optimized-prioritized allocation strategy

The following [create-auto-scaling-group](#) command creates an Auto Scaling group that specifies the following:

- The percentage of the group to launch as On-Demand Instances (0) and a base number of On-Demand Instances to start with (1)
- The instance types to launch in priority order (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large)
- The subnets in which to launch the instances (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782), each corresponding to a different Availability Zone
- The launch template (`my-launch-template`) and the launch template version (`$Latest`)

When Amazon EC2 Auto Scaling attempts to fulfill your On-Demand capacity, it launches the c5.large instance type first. When Amazon EC2 Auto Scaling attempts to fulfill your Spot capacity, it honors the instance type priorities on a best-effort basis, but optimizes for capacity first.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                }
            ]
        }
    }
}
```

```

        },
        {
            "InstanceType": "m3.large"
        }
    ],
    "InstancesDistribution": {
        "OnDemandBaseCapacity": 1,
        "OnDemandPercentageAboveBaseCapacity": 0,
        "SpotAllocationStrategy": "capacity-optimized-prioritized"
    }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}

```

YAML

Alternatively, you can use the following [create-auto-scaling-group](#) command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

The following is an example config.yaml file.

```

---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceType: c5.large
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
  MinSize: 1
  MaxSize: 5
  DesiredCapacity: 3
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782

```

Example 3: Launch Spot Instances using the lowest-price allocation strategy diversified over two pools

The following [create-auto-scaling-group](#) command creates an Auto Scaling group that specifies the following:

- The percentage of the group to launch as On-Demand Instances (50) without also specifying a base number of On-Demand Instances to start with

- The instance types to launch in priority order (c5.large, c5a.large, m5.large, m5a.large, c4.large, m4.large, c3.large, m3.large)
- The subnets in which to launch the instances (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782), each corresponding to a different Availability Zone
- The launch template (my-launch-template) and the launch template version (\$Latest)

When Amazon EC2 Auto Scaling attempts to fulfill your On-Demand capacity, it launches the c5.large instance type first. For your Spot capacity, Amazon EC2 Auto Scaling attempts to launch the Spot Instances evenly across the two lowest-priced pools in each Availability Zone.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        },
        {
          "InstanceType": "m5.large"
        },
        {
          "InstanceType": "m5a.large"
        },
        {
          "InstanceType": "c4.large"
        },
        {
          "InstanceType": "m4.large"
        },
        {
          "InstanceType": "c3.large"
        },
        {
          "InstanceType": "m3.large"
        }
      ]
    },
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "lowest-price",
      "SpotInstancePools": 2
    }
  },
  "MinSize": 1,
  "MaxSize": 5,
  "DesiredCapacity": 3,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

}

YAML

Alternatively, you can use the following [create-auto-scaling-group](#) command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

The following is an example config.yaml file.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
  Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
  MinSize: 1
  MaxSize: 5
  DesiredCapacity: 3
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Note

For additional examples, see [Specify a different launch template for an instance type \(p. 65\)](#), [Configure instance weighting for Amazon EC2 Auto Scaling \(p. 68\)](#), and [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).

Verify whether your Auto Scaling group is configured correctly and that the group has launched instances (AWS CLI)

To check whether your Auto Scaling group is configured correctly and that it has launched instances, use the [describe-auto-scaling-groups](#) command. Verify that the mixed instances policy and the list of subnets exist and are configured correctly. If the instances have launched, you will see a list of the instances and their statuses. To view the scaling activities that result from instances launching, use the [describe-scaling-activities](#) command. You can monitor scaling activities that are in progress and that are recently completed.

Configure overrides

Overrides are changes you make to the properties of your launch template. Amazon EC2 Auto Scaling supports overrides to the instance type property. That way, you can specify multiple instance types. To do so, you must add the Overrides structure to your mixed instances policy.

The **Overrides** structure allows you to define a set of parameters that Amazon EC2 Auto Scaling can use to launch instances, including:

- **InstanceType** — The instance type. For more information about the instance types that are available, see [Instance types](#) in the *Amazon EC2 User Guide for Linux Instances*.
- **LaunchTemplateSpecification** — (Optional) A launch template for an individual instance type. This property is currently limited to the AWS CLI or an SDK, and is not available from the console.
- **WeightedCapacity** — (Optional) The number of units that a provisioned instance of this type provides toward fulfilling the desired capacity of the Auto Scaling group. If you specify a weight value for one instance type, you must specify a weight value for all of them.

Note

As an alternative to specifying a list of instance types, you can specify a set of instance attributes to use as criteria for selecting the instance types your Auto Scaling group uses. This is known as *attribute-based instance type selection*. For more information, see [Using attribute-based instance type selection \(p. 75\)](#).

Contents

- [Specify a different launch template for an instance type \(p. 65\)](#)
- [Configure instance weighting for Amazon EC2 Auto Scaling \(p. 68\)](#)

Specify a different launch template for an instance type

The **Overrides** structure allows you to define a new launch template for individual instance types for a new or existing Auto Scaling group. For example, if the architecture of an instance type requires a different AMI from the rest of the group, you must specify a launch template with a compatible AMI.

Say that you configure an Auto Scaling group for compute-intensive applications and want to include a mix of C5, C5a, and C6g instance types. However, C6g instances feature an AWS Graviton processor based on 64-bit Arm architecture, while the C5 and C5a instances run on 64-bit Intel x86 processors. The AMI for C5 instances works on C5a instances and vice-versa, but not on C6g instances. The **Overrides** property allows you to include a different launch template for C6g instances, while still using the same launch template for C5 and C5a instances.

Note

Currently, this feature is available only if you use the AWS CLI or an SDK, and is not available from the console.

Contents

- [Add or change a launch template for an instance type \(AWS CLI\) \(p. 65\)](#)

Add or change a launch template for an instance type (AWS CLI)

The following procedure shows you how to use the AWS CLI to configure an Auto Scaling group so that one or more of the instance types uses a launch template that is different from the rest of the group.

To create and configure a new Auto Scaling group

1. Create a configuration file where you specify a mixed instances policy structure and include the **Overrides** structure.

The following are the contents of an example configuration file formatted in JSON. It specifies the `c5.large`, `c5a.large`, and `c6g.large` instance types and defines a new launch template for the `c6g.large` instance type to ensure that an appropriate AMI is used to launch Arm instances.

Amazon EC2 Auto Scaling uses the order of instance types to determine which instance type to use first when fulfilling On-Demand capacity.

```
{
  "AutoScalingGroupName": "my-asg",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template-for-x86",
        "Version": "$Latest"
      },
      "Overrides": [
        {
          "InstanceType": "c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
        {
          "InstanceType": "c5.large"
        },
        {
          "InstanceType": "c5a.large"
        }
      ]
    },
    "InstancesDistribution": {
      "OnDemandBaseCapacity": 1,
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize": 1,
  "MaxSize": 5,
  "DesiredCapacity": 3,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags": []
}
```

2. Use the following [create-auto-scaling-group](#) command, referencing the JSON file as the sole parameter for your Auto Scaling group.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

To change the launch template for an instance type in an existing Auto Scaling group

- Use the following [update-auto-scaling-group](#) command to specify a different launch template for an instance type by passing the Overrides structure.

When this change is made, any new instances that are launched are based on the new settings, but existing instances are not affected. To ensure that your Auto Scaling group is using the new settings, you can replace all instances in the group by starting an instance refresh or by using the maximum instance lifetime feature.

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{  
  "AutoScalingGroupName": "my-asg",  
  "MixedInstancesPolicy": {  
    "LaunchTemplate": {  
      "Overrides": [  
        {  
          "InstanceType": "c6g.large",  
          "LaunchTemplateSpecification": {  
            "LaunchTemplateName": "my-launch-template-for-arm",  
            "Version": "$Latest"  
          }  
        },  
        {  
          "InstanceType": "c5.large"  
        },  
        {  
          "InstanceType": "c5a.large"  
        }  
      ]  
    }  
  }  
}
```

To verify the launch templates for an Auto Scaling group

- Use the following [describe-auto-scaling-groups](#) command to verify and view the currently specified launch templates.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn",  
      "MixedInstancesPolicy": {  
        "LaunchTemplate": {  
          "LaunchTemplateSpecification": {  
            "LaunchTemplateId": "lt-0fb0e487336917fb2",  
            "LaunchTemplateName": "my-launch-template-for-x86",  
            "Version": "$Latest"  
          }  
        },  
        "Overrides": [  
          {  
            "InstanceType": "c6g.large",  
            "LaunchTemplateSpecification": {  
              "LaunchTemplateId": "lt-09d958b8fb2ba5bcc",  
              "LaunchTemplateName": "my-launch-template-for-arm",  
              "Version": "$Latest"  
            }  
          },  
          {  
            "InstanceType": "c5.large"  
          },  
          {  
            "InstanceType": "c5a.large"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```

        },
        "InstancesDistribution": {
            "OnDemandAllocationStrategy": "prioritized",
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "Instances": [
        {
            "InstanceId": "i-07c63168522c0f620",
            "InstanceType": "c5.large",
            "AvailabilityZone": "us-west-2c",
            "LifecycleState": "InService",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0fb0e487336917fb2",
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "1"
            },
            "ProtectedFromScaleIn": false
        },
        {
            "InstanceId": "i-0b7ff78be9896a2c2",
            "InstanceType": "c5.large",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "InService",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0fb0e487336917fb2",
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "1"
            },
            "ProtectedFromScaleIn": false
        },
        {
            "InstanceId": "i-0c682c2ceae918bc0",
            "InstanceType": "c6g.large",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-09d958b8fb2ba5bcc",
                "LaunchTemplateName": "my-launch-template-for-arm",
                "Version": "1"
            },
            ...
        }
    ]
}

```

Configure instance weighting for Amazon EC2 Auto Scaling

When you configure an Auto Scaling group to launch multiple instance types, you have the option of defining the number of capacity units that each instance contributes to the desired capacity of the group, using *instance weighting*. This allows you to specify the relative weight of each instance type in a way that directly maps to the performance of your application. You can weight your instances to suit your specific application needs, for example, by the cores (vCPUs) or by memory (GiBs).

For example, let's say that you run a compute-intensive application that performs best with at least 8 vCPUs and 15 GiB of RAM. If you use `c5.2xlarge` as your base unit, any of the following EC2 instance types would meet your application needs.

Instance types example

Instance type	vCPU	Memory (GiB)
<code>c5.2xlarge</code>	8	16
<code>c5.4xlarge</code>	16	32
<code>c5.12xlarge</code>	48	96
<code>c5.18xlarge</code>	72	144
<code>c5.24xlarge</code>	96	192

By default, all instance types are treated as the same weight. In other words, whether Amazon EC2 Auto Scaling launches a large or small instance type, each instance counts toward the group's desired capacity.

With instance weighting, however, you assign a number value that specifies how many capacity units to associate with each instance type. For example, if the instances are of different sizes, a `c5.2xlarge` instance could have the weight of 2, and a `c5.4xlarge` (which is two times bigger) could have the weight of 4, and so on. Then, when Amazon EC2 Auto Scaling launches instances, their weights count toward your desired capacity.

Contents

- [Price per unit hour \(p. 69\)](#)
- [Considerations \(p. 70\)](#)
- [Add or modify weights for your Auto Scaling group \(p. 71\)](#)
- [Additional information \(p. 74\)](#)

Price per unit hour

The following table compares the hourly price for Spot Instances in different Availability Zones in US East (N. Virginia, Ohio) with the price for On-Demand Instances in the same Region. The prices shown are example pricing and not current pricing. These are your costs *per instance hour*.

Example: Spot pricing per instance hour

Instance type	us-east-1a	us-east-1b	us-east-1c	On-Demand pricing
<code>c5.2xlarge</code>	\$0.180	\$0.191	\$0.170	\$0.34
<code>c5.4xlarge</code>	\$0.341	\$0.361	\$0.318	\$0.68
<code>c5.12xlarge</code>	\$0.779	\$0.777	\$0.777	\$2.04
<code>c5.18xlarge</code>	\$1.207	\$1.475	\$1.357	\$3.06
<code>c5.24xlarge</code>	\$1.555	\$1.555	\$1.555	\$4.08

With instance weighting, you can evaluate your costs based on what you use *per unit hour*. You can determine the price per unit hour by dividing your price for an instance type by the number of units that

it represents. For On-Demand Instances, the price *per unit hour* is the same when deploying one instance type as it is when deploying a different size of the same instance type. In contrast, however, the Spot price *per unit hour* varies by Spot pool.

The easiest way to understand how the price *per unit hour* calculation works with weighted instances is with an example. For example, for ease of calculation, let's say you want to launch Spot Instances only in us-east-1a. The *per unit hour* price is captured below.

Example: Spot Price per unit hour example

Instance type	us-east-1a	Instance weight	Price per unit hour
c5.2xlarge	\$0.180	2	\$0.090
c5.4xlarge	\$0.341	4	\$0.085
c5.12xlarge	\$0.779	12	\$0.065
c5.18xlarge	\$1.207	18	\$0.067
c5.24xlarge	\$1.555	24	\$0.065

Considerations

This section discusses the key considerations in implementing instance weighting effectively.

- Start by choosing a few instance types that reflect the actual performance requirements of your application. Then, decide how much each instance type should count toward the desired capacity of your Auto Scaling group by specifying their weights. The weights apply to current and future instances in the group.
- Be cautious about choosing very large ranges for your weights. For example, we don't recommend specifying a weight of 1 for an instance type when the next larger instance type has a weight of 200. The difference between the smallest and largest weights should also not be extreme. If any of the instance types have too large of a weight difference, this can have a negative effect on ongoing cost-performance optimization.
- The size of the Auto Scaling group is measured in capacity units, and not in instances. For example, if your weights are based on vCPUs, you must specify the desired, minimum, and maximum number of cores you want.
- Set your weights and desired capacity so that the desired capacity is at least two to three times larger than your largest weight.

With instance weighting, the following new behaviors are introduced:

- Current capacity will either be at the desired capacity or above it. Because Amazon EC2 Auto Scaling wants to provision instances until the desired capacity is totally fulfilled, an overage can happen. For example, suppose that you specify two instance types, c5.2xlarge and c5.12xlarge, and you assign instance weights of 2 for c5.2xlarge and 12 for c5.12xlarge. If there are 5 units remaining to fulfill the desired capacity, and Amazon EC2 Auto Scaling provisions a c5.12xlarge, the desired capacity is exceeded by 7 units.
- When Amazon EC2 Auto Scaling provisions instances to reach the desired capacity, distributing instances across Availability Zones and respecting the allocation strategies for On-Demand and Spot Instances both take precedence over avoiding overages.
- Amazon EC2 Auto Scaling can overstep the maximum capacity limit to maintain balance across Availability Zones, using your preferred allocation strategies. The hard limit enforced by Amazon EC2 Auto Scaling is a value that is equal to your desired capacity plus your largest weight.

Note the following when adding or modifying weights for existing groups:

- When adding instance weights to an existing Auto Scaling group, you must include any instance types that are already running in the group.
- When modifying existing instance weights, Amazon EC2 Auto Scaling will launch or terminate instances to reach your desired capacity based on the new weights.
- If you remove an instance type, any running instances of that instance type will continue to have their last updated weight values, even though the instance type has been removed.

Add or modify weights for your Auto Scaling group

You can add weights to an existing Auto Scaling group, or to a new Auto Scaling group as you create it. You can also update an existing Auto Scaling group to define new configuration options (Spot/On-Demand usage, Spot allocation strategy, instance types). If you change how many Spot or On-Demand Instances you want, Amazon EC2 Auto Scaling gradually replaces existing instances to match the new purchase options.

Before creating Auto Scaling groups using instance weighting, we recommend that you become familiar with launching groups with multiple instance types. For more information and additional examples, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

The following examples show how to use the AWS CLI to add weights when you create Auto Scaling groups, and to add or modify weights for existing Auto Scaling groups. You can configure a variety of parameters in a JSON file, and then reference the JSON file as the sole parameter for your Auto Scaling group.

To add weights to an Auto Scaling group on creation

- Use the [create-auto-scaling-group](#) command to create a new Auto Scaling group. For example, the following command creates a new Auto Scaling group and adds instance weighting by specifying the following:
 - The percentage of the group to launch as On-Demand Instances (0)
 - The allocation strategy for Spot Instances in each Availability Zone (capacity-optimized)
 - The instance types to launch in priority order (m4.16xlarge, m5.24xlarge)
 - The instance weights that correspond to the relative size difference (vCPUs) between instance types (16, 24)
 - The subnets in which to launch the instances (subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782), each corresponding to a different Availability Zone
 - The launch template (my-launch-template) and the launch template version (\$Latest)

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "MixedInstancesPolicy": {  
        "LaunchTemplate": {  
            "LaunchTemplateSpecification": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "$Latest"  
            },  
            "Overrides": [  
                {  
                    "InstanceType": "m4.16xlarge",  
                    "Weight": 16  
                },  
                {  
                    "InstanceType": "m5.24xlarge",  
                    "Weight": 24  
                }  
            ]  
        }  
    }  
}
```

```

        "InstanceType": "m4.16xlarge",
        "WeightedCapacity": "16"
    },
    {
        "InstanceType": "m5.24xlarge",
        "WeightedCapacity": "24"
    }
],
"InstancesDistribution": {
    "OnDemandPercentageAboveBaseCapacity": 0,
    "SpotAllocationStrategy": "capacity-optimized"
}
},
"MinSize": 160,
"MaxSize": 720,
"DesiredCapacity": 480,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
"Tags": []
}

```

To add or modify weights for an existing Auto Scaling group

- Use the [update-auto-scaling-group](#) command to add or modify weights. For example, the following command adds weights to instance types in an existing Auto Scaling group by specifying the following:
 - The instance types to launch in priority order (c5.18xlarge, c5.24xlarge, c5.2xlarge, c5.4xlarge)
 - The instance weights that correspond to the relative size difference (vCPUs) between instance types (18, 24, 2, 4)
 - The new, increased desired capacity, which is larger than the largest weight

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{
    "AutoScalingGroupName": "my-existing-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "Overrides": [
                {
                    "InstanceType": "c5.18xlarge",
                    "WeightedCapacity": "18"
                },
                {
                    "InstanceType": "c5.24xlarge",
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceType": "c5.2xlarge",
                    "WeightedCapacity": "2"
                },
                {
                    "InstanceType": "c5.4xlarge",
                    "WeightedCapacity": "4"
                }
            ]
        }
    }
}
```

```

        },
        "MinSize": 0,
        "MaxSize": 100,
        "DesiredCapacity": 100
    }
}
```

To verify the weights for an Auto Scaling group

- Use the following [describe-auto-scaling-groups](#) command to verify the weights.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "$Latest"
                    },
                    "Overrides": [
                        {
                            "InstanceType": "m4.16xlarge",
                            "WeightedCapacity": "16"
                        },
                        {
                            "InstanceType": "m5.24xlarge",
                            "WeightedCapacity": "24"
                        }
                    ]
                },
                "InstancesDistribution": {
                    "OnDemandAllocationStrategy": "prioritized",
                    "OnDemandBaseCapacity": 0,
                    "OnDemandPercentageAboveBaseCapacity": 0,
                    "SpotAllocationStrategy": "capacity-optimized"
                }
            },
            "MinSize": 160,
            "MaxSize": 720,
            "DesiredCapacity": 480,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-027327f0ace86f499",
                    "InstanceType": "m5.24xlarge",

```

```
        "AvailabilityZone": "us-west-2a",
        "LifecycleState": "InService",
        "HealthStatus": "Healthy",
        "LaunchTemplate": {
            "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
            "LaunchTemplateName": "my-launch-template",
            "Version": "7"
        },
        "ProtectedFromScaleIn": false,
        "WeightedCapacity": "24"
    },
    {
        "InstanceId": "i-0ec0d761cc134878d",
        "InstanceType": "m4.16xlarge",
        "AvailabilityZone": "us-west-2a",
        "LifecycleState": "Pending",
        "HealthStatus": "Healthy",
        "LaunchTemplate": {
            "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
            "LaunchTemplateName": "my-launch-template",
            "Version": "7"
        },
        "ProtectedFromScaleIn": false,
        "WeightedCapacity": "16"
    },
    ...
]
}
```

Additional information

Instance weighting and allocation strategies

The [allocation strategies \(p. 53\)](#) determine which instance pools your instances come from. When you use the instance weighting feature, the allocation strategies perform exactly like they do for other Auto Scaling groups. However, there is one key difference in how instance pools are chosen when you use the **lowest-price** strategy. When you choose **lowest-price** for your allocation strategy, your instances come from the instance pools with the lowest price per unit in each Availability Zone.

For example, consider you have an Auto Scaling group that has several instance types with varying amounts of vCPUs. You use **lowest-price** for your Spot and On-Demand allocation strategies. If you choose to assign weights based on the vCPU count of each instance type, Amazon EC2 Auto Scaling launches whichever instance types have the lowest price per your assigned weight values (for example, per vCPU) at the time of fulfillment. If it's a Spot Instance, then this means the lowest Spot price per vCPU. If it's an On-Demand Instance, then this means the lowest On-Demand price per vCPU.

Instance weighting and Spot max price

When you create your Auto Scaling group using the AWS CLI or an SDK, you can specify the `SpotMaxPrice` parameter. This parameter determines the maximum price you would be willing to pay for a Spot instance hour, and by default, is set at the On-Demand price. When you use the instance weighting feature, remember that the maximum Spot price reflects the maximum unit price (for example, price per vCPU) instead of the maximum price for a whole instance.

Create an Auto Scaling group using attribute-based instance type selection

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the Availability Zones and VPC subnets for the instances, the desired capacity, and the minimum and maximum capacity limits.

As an alternative to manually choosing instance types when creating a [mixed instances group \(p. 52\)](#), you can specify a set of instance attributes that describe your compute requirements. As Amazon EC2 Auto Scaling launches instances, any instance types used by the Auto Scaling group must match your required instance attributes. This is known as *attribute-based instance type selection*.

Your Auto Scaling group or your launch template specifies your instance attributes, including the amount of memory and computing power that you need for the applications that you plan to run on the instances. Additionally, your Auto Scaling group or your launch template specifies two price protection thresholds for Spot and On-Demand Instances, that you can optionally customize, so that you can prevent Amazon EC2 Auto Scaling from launching more expensive instance types if you don't need them.

This approach is ideal for workloads and frameworks that can be flexible about which instance types they use, such as containers, big data, and CI/CD.

The following are benefits of attribute-based instance type selection:

- Amazon EC2 Auto Scaling can select from a wide range of instance types for launching Spot Instances. This meets the Spot best practice of being flexible about instance types, which gives the Amazon EC2 Spot service a better chance of finding and allocating your required amount of compute capacity.
- With so many options available, finding the right instance types for your workload can be time consuming. By specifying instance attributes, you can simplify instance type selection when configuring a mixed instances group.
- Your Auto Scaling groups can use newer generation instance types as they're released. Newer generation instance types are automatically used when they match your requirements and align with the allocation strategies you choose for your Auto Scaling group.

You can use attribute-based instance type selection through the AWS Management Console, AWS CLI, or SDKs.

For information about how to configure attribute-based instance type selection in a launch template, see [Create a launch template for an Auto Scaling group \(p. 22\)](#). For information about how to configure attribute-based instance type selection by passing parameters in Amazon EC2 Auto Scaling API calls using an SDK, see [InstanceRequirements](#) in the [Amazon EC2 Auto Scaling API Reference](#).

To learn more about attribute-based instance type selection, see [Attribute-Based Instance Type Selection for EC2 Auto Scaling and EC2 Fleet](#) on the AWS Blog.

Contents

- [Considerations \(p. 75\)](#)
- [Prerequisites \(p. 76\)](#)
- [Use attribute-based instance type selection \(p. 77\)](#)
- [Limitations \(p. 80\)](#)

Considerations

The following are things to consider when using attribute-based instance type selection:

- For most general purpose workloads, it is enough to specify the number of vCPUs and memory that you need. For advanced use cases, you can specify attributes like storage type, network interfaces, CPU manufacturer, and accelerator type.
- By default, you set the value for the desired capacity of your Auto Scaling group as the number of instances. You can optionally specify the desired capacity type as the number of vCPUs or the amount of memory when you use attribute-based instance type selection. Then, when Amazon EC2 Auto Scaling launches instances, their number of vCPUs or amount of memory count toward your desired capacity. When creating your group in the Amazon EC2 Auto Scaling console, this setting appears in the **Group size** section on the **Configure group size and scaling policies** page. This feature is a useful replacement for the [instance weighting \(p. 68\)](#) feature.
- You can preview the instance types that match your compute requirements without launching them and adjust your requirements if necessary. When creating your Auto Scaling group in the Amazon EC2 Auto Scaling console, a preview of the instance types appears in the **Preview matching instance types** section on the **Choose instance launch options** page.
- Alternatively, you can preview the instance types by making an Amazon EC2 [GetInstanceTypesFromInstanceRequirements](#) API call using the AWS CLI or an SDK. Pass the `InstanceRequirements` parameters in the request in the exact format that you would use to create or update an Auto Scaling group. For more information, see [Preview instance types with specified attributes in the Amazon EC2 User Guide for Linux Instances](#).

Understand price protection

Price protection is a feature that protects your Auto Scaling group from extreme price differences across instance types. When you create a new Auto Scaling group or update an existing Auto Scaling group with attribute-based instance type selection, we enable price protection by default. You can optionally choose your price protection thresholds for Spot and On-Demand Instances, and Amazon EC2 Auto Scaling doesn't select instance types whose price is higher than your specified thresholds. The thresholds represent what you are willing to pay, defined in terms of a percentage above a baseline, rather than as absolute values. The baseline is determined by the price of the least expensive current generation M, C, or R instance type with your specified attributes. If your attributes don't match any M, C, or R instance types, we use the lowest priced instance type.

If you don't specify a threshold, the following are used by default:

- For On-Demand Instances, the price protection threshold is set to 20 percent.
- For Spot Instances, the price protection threshold is set to 100 percent.

To update these values when creating your Auto Scaling group in the Amazon EC2 Auto Scaling console, on the **Choose instance launch options** page, choose the desired price protection attribute from the **Additional instance attributes** drop-down list, and then type or choose a value for the attribute in the text box. You can also update these values at any time by editing the Auto Scaling group from the console or passing the relevant parameters using the AWS CLI or an SDK.

Note

If you set **Desired capacity type** to **vCPUs** or **Memory GiB**, the price protection threshold is applied based on the per vCPU or per memory price instead of the per instance price.

Prerequisites

Create a launch template that includes the parameters required to launch an EC2 instance, such as the Amazon Machine Image (AMI) and security groups. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

Verify that you have the permissions required to use a launch template. Your `ec2:RunInstances` permissions are checked when use a launch template. Your `iam:PassRole` permissions are also

checked if the launch template specifies an IAM role. For more information, see [Launch template support \(p. 347\)](#).

Use attribute-based instance type selection

The following steps describe how to create an Auto Scaling group using attribute-based instance type selection:

- Specify the launch template from which to launch the instances.
- Choose which VPC and subnets you are launching your Auto Scaling group in.
- Choose the option to override the existing instance type requirements of the launch template with new requirements.
- Specify instance attributes that match your compute requirements, such as vCPUs, memory, and storage.
- Specify the percentages of On-Demand Instances and Spot Instances to launch.
- Choose allocation strategies that determine how Amazon EC2 Auto Scaling fulfills your On-Demand and Spot capacities from the possible instance types.
- Specify your group size, including your minimum capacity, maximum capacity, desired capacity, and unit of measurement.

Before you begin, confirm that you have created a launch template, as described in [Prerequisites \(p. 76\)](#).

Verify that the launch template doesn't already request Spot Instances.

To create an Auto Scaling group using attribute-based instance type selection (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. On the navigation bar at the top of the screen, choose the same AWS Region that you used when you created the launch template.
3. Choose **Create an Auto Scaling group**.
4. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.
5. To choose your launch template, do the following:
 - a. For **Launch template**, choose an existing launch template.
 - b. For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.
 - c. Verify that your launch template supports all of the options that you are planning to use, and then choose **Next**.
6. To choose a VPC and the subnets you want to launch your instances in, do the following:
 - a. On the **Choose instance launch options** page, under **Network**, for **VPC**, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch template.
 - b. For **Availability Zones and subnets**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).
7. To configure your group to use attribute-based instance type selection, do the following:
 - a. For **Instance type requirements**, choose **Override launch template**.

Note

If you chose a launch template that already contains your requirements, the settings that you defined in your launch template, such as vCPUs and memory, are automatically populated as the attributes to use. You can update these attributes from the Amazon EC2 Auto Scaling console as needed at any time.

- b. Under **Specify instance attributes**, start by entering your vCPUs and memory requirements.
 - **vCPUs**: Enter the minimum and maximum number of vCPUs for your compute requirements. Select the **No minimum** or **No maximum** check box to indicate no limit in that direction.
 - **Memory (MiB)**: Enter the minimum and maximum amount of memory, in MiB, for your compute requirements. Select the **No minimum** or **No maximum** check box to indicate no limit in that direction.
- c. (Optional) For **Additional instance attributes**, choose **Add attribute** to express your compute requirements in more detail. The attributes and values that you choose here further define which instance types may be launched.

For more information about all the supported attributes, see [InstanceRequirements](#) in the *Amazon EC2 Auto Scaling API Reference*.

- d. For **Preview matching instance types**, view the instance types that match the specified compute requirements, such as vCPUs, memory, and storage.
- e. Under **Instance purchase options**, for **Instances distribution**, specify the percentages of the group to be launched as On-Demand Instances and Spot Instances respectively. If your application is stateless, fault tolerant and can handle an instance being interrupted, you can specify a higher percentage of Spot Instances.
- f. When you specify a percentage for Spot Instances, you can select the check box next to **Include On-Demand base capacity** and then specify the minimum amount of the Auto Scaling group's initial capacity that must be fulfilled by On-Demand Instances. Anything beyond the base capacity uses the **Instances distribution** settings to determine how many On-Demand Instances and Spot Instances to launch.
- g. Under **Allocation strategies**, **Lowest price** is automatically selected for the **On-Demand allocation strategy** and cannot be changed.
- h. For **Spot allocation strategy**, choose an allocation strategy. We recommend that you keep the default setting of **Capacity optimized**. If you prefer not to keep the default, choose **Lowest price**, and then enter the number of lowest priced Spot Instance pools to diversify across.
- i. For **Capacity rebalance**, choose whether to enable or disable Capacity Rebalancing.

If you chose a percentage for Spot Instances, you can use Capacity Rebalancing to respond through automation to the scenario where your Spot Instances are approaching termination from a Spot interruption. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).

- j. When finished, choose **Next** twice to go to the **Configure group size and scaling policies** page.
8. For the **Configure group size and scaling policies** step, do the following:
 - a. If you want your desired capacity to be measured in units other than instances, choose the appropriate option for **Desired capacity type**. **Units**, **vCPUs** and **Memory GiB** are supported. By default, Amazon EC2 Auto Scaling specifies **Units**, which translates into number of instances.
 - b. Enter the initial size of your Auto Scaling group for **Desired capacity** and update the **Minimum capacity** and **Maximum capacity** limits as needed. For more information, see [Set capacity limits on your Auto Scaling group \(p. 122\)](#).
 - c. (Optional) To configure the group to scale using a target tracking scaling policy, you have the option to specify the scaling policy now, or after the group has been created. For more information, see [Target tracking scaling policies for Amazon EC2 Auto Scaling \(p. 135\)](#).

- d. (Optional) To prevent your Auto Scaling group from terminating instances when scaling in, enable instance scale-in protection. For more information, see [Use instance scale-in protection \(p. 240\)](#).
- e. When finished, choose **Next**.
9. (Optional) To receive notifications when your group scales, for **Add notification**, configure the notification, and then choose **Next**. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).
10. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see [Tag Auto Scaling groups and instances \(p. 91\)](#).
11. On the **Review** page, choose **Create Auto Scaling group**.

Example: Create an Auto Scaling group using attribute-based instance type selection (AWS CLI)

To create an Auto Scaling group with attribute-based instance type selection using the command line, you can use the following [create-auto-scaling-group](#) command.

The following instance attributes are specified:

- **VCpuCount** – The instance types must have a minimum of four vCPUs and a maximum of eight vCPUs.
- **MemoryMiB** – The instance types must have a minimum of 16384 MiB of memory.
- **CpuManufacturers** – The instance types must have an Intel manufactured CPU.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredCapacityType": "units",
  "MixedInstancesPolicy": {
    "LaunchTemplate": {
      "LaunchTemplateSpecification": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
      },
      "Overrides": [
        {
          "InstanceRequirements": {
            "VCpuCount": {"Min": 4, "Max": 8},
            "MemoryMiB": {"Min": 16384},
            "CpuManufacturers": ["intel"]
          }
        }
      ]
    },
    "InstancesDistribution": {
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize": 0,
  "MaxSize": 100,
  "DesiredCapacity": 4,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

}

To set the value for desired capacity as the number of vCPUs or the amount of memory, specify "DesiredCapacityType": "vcpu" or "DesiredCapacityType": "memory-mib" in the file. The default desired capacity type is units, which sets the value for desired capacity as the number of instances.

YAML

Alternatively, you can use the following [create-auto-scaling-group](#) command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

The following is an example config.yaml file.

```
---
AutoScalingGroupName: my-asg
DesiredCapacityType: units
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
      - InstanceRequirements:
        VCpuCount:
          Min: 2
          Max: 4
        MemoryMiB:
          Min: 2048
        CpuManufacturers:
          - intel
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: capacity-optimized
  MinSize: 0
  MaxSize: 100
  DesiredCapacity: 4
  VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

To set the value for desired capacity as the number of vCPUs or the amount of memory, specify DesiredCapacityType: vcpu or DesiredCapacityType: memory-mib in the file. The default desired capacity type is units, which sets the value for desired capacity as the number of instances.

Limitations

- You can configure attribute-based instance type selection only for Auto Scaling groups that use a launch template.
- If you have an existing Auto Scaling group and plan to substitute your instance types with your required instance attributes, your On-Demand allocation strategy must be lowest-price. To use the prioritized allocation strategy, you must continue to manually add and prioritize your instance types. Additionally, your Spot allocation strategy must be either capacity-optimized or lowest-price. To use the capacity-optimized-prioritized allocation strategy, you must manually add and prioritize your instance types.

Create an Auto Scaling group using a launch template

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the Availability Zones and VPC subnets for the instances, the desired capacity, and the minimum and maximum capacity limits.

To configure Amazon EC2 instances that are launched by your Auto Scaling group, you can specify a launch template or a launch configuration. The following procedure demonstrates how to create an Auto Scaling group using a launch template.

To update the configuration of the EC2 instances after the group is created, you can create a new version of the launch template. After you change the launch template for an Auto Scaling group, any new instances are launched using the new configuration options, but existing instances are not affected. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow auto scaling to gradually replace older instances with newer instances based on your [termination policies \(p. 230\)](#).

Note

You can also replace all instances in the Auto Scaling group to launch new instances that use the new launch template. For more information, see [Replace Auto Scaling instances \(p. 99\)](#).

Prerequisites

- You must have created a launch template. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).
- You must have IAM permissions to create an Auto Scaling group using a launch template. Your `ec2:RunInstances` permissions are checked when use a launch template. Your `iam:PassRole` permissions are also checked if the launch template specifies an IAM role. For more information, see [Launch template support \(p. 347\)](#).

To create an Auto Scaling group using a launch template (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. On the navigation bar at the top of the screen, choose the same AWS Region that you used when you created the launch template.
3. Choose **Create an Auto Scaling group**.
4. On the **Choose launch template or configuration** page, do the following:
 - a. For **Auto Scaling group name**, enter a name for your Auto Scaling group.
 - b. For **Launch template**, choose an existing launch template.
 - c. For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.
 - d. Verify that your launch template supports all of the options that you are planning to use, and then choose **Next**.
5. On the **Choose instance launch options** page, under **Network**, for **VPC**, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch template.
6. For **Availability Zones and subnets**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).

7. If you created a launch template with an instance type specified, then you can continue to the next step to create an Auto Scaling group that uses the instance type in the launch template.

Alternatively, you can choose the **Override launch template** option if no instance type is specified in your launch template or if you want to use multiple instance types for auto scaling. For more information, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

8. Choose **Next** to continue to the next step.

Or, you can accept the rest of the defaults, and choose **Skip to review**.

9. (Optional) On the **Configure advanced options** page, configure the following options, and then choose **Next**:

- To register your Amazon EC2 instances with a load balancer, choose an existing load balancer or create a new one. For more information, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#). To create a new load balancer, follow the procedure in [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#).
- To enable your Elastic Load Balancing (ELB) health checks, for **Health checks**, choose **ELB** under **Health check type**. These health checks are optional when you enable load balancing.
- Under **Health check grace period**, enter the amount of time until Amazon EC2 Auto Scaling checks the Elastic Load Balancing health status of an instance after it enters the **InService** state. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).
- Under **Additional settings, Monitoring**, choose whether to enable CloudWatch group metrics collection. These metrics provide measurements that can be indicators of a potential issue, such as number of terminating instances or number of pending instances. For more information, see [Monitor CloudWatch metrics for your Auto Scaling groups and instances \(p. 260\)](#).
- For **Enable default instance warmup**, select this option and choose the warm-up time for your application. If you are creating an Auto Scaling group that has a scaling policy, the default instance warmup feature improves the Amazon CloudWatch metrics used for dynamic scaling. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).

10. (Optional) On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**:

- For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see [Set capacity limits on your Auto Scaling group \(p. 122\)](#).
- To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see [Target tracking scaling policies for Amazon EC2 Auto Scaling \(p. 135\)](#).
- Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see [Use instance scale-in protection \(p. 240\)](#).

11. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).

12. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see [Tag Auto Scaling groups and instances \(p. 91\)](#).

13. On the **Review** page, choose **Create Auto Scaling group**.

To create an Auto Scaling group using the command line

You can use one of the following commands:

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

Create an Auto Scaling group using a launch configuration

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the Availability Zones and VPC subnets for the instances, the desired capacity, and the minimum and maximum capacity limits.

Important

To configure the Amazon EC2 instances, you can specify a launch template or a launch configuration. We recommend that you use a launch template to make sure that you can use the latest features of Amazon EC2 Auto Scaling and Amazon EC2. For more information, see [Create an Auto Scaling group using a launch template \(p. 81\)](#).

The following procedure demonstrates how to create an Auto Scaling group using a launch configuration. You cannot modify a launch configuration after it is created, but you can replace the launch configuration for an Auto Scaling group. For more information, see [Change the launch configuration for an Auto Scaling group \(p. 47\)](#).

Prerequisites

You must have created a launch configuration. For more information, see [Create a launch configuration \(p. 40\)](#).

To create an Auto Scaling group using a launch configuration (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. On the navigation bar at the top of the screen, choose the same AWS Region that you used when you created the launch configuration.
 3. Choose **Create an Auto Scaling group**.
 4. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.
 5. To choose a launch configuration, do the following:
 - a. For **Launch template**, choose **Switch to launch configuration**.
 - b. For **Launch configuration**, choose an existing launch configuration.
 - c. Verify that your launch configuration supports all of the options that you are planning to use, and then choose **Next**.
 6. On the **Configure instance launch options** page, under **Network**, for **VPC**, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch configuration.
 7. For **Availability Zones and subnets**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).
 8. Choose **Next**.
- Or, you can accept the rest of the defaults, and choose **Skip to review**.
9. (Optional) On the **Configure advanced options** page, configure the following options, and then choose **Next**:

- a. To register your Amazon EC2 instances with a load balancer, choose an existing load balancer or create a new one. For more information, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#). To create a new load balancer, follow the procedure in [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#).
 - b. To enable your Elastic Load Balancing (ELB) health checks, for **Health checks**, choose **ELB** under **Health check type**. These health checks are optional when you enable load balancing.
 - c. Under **Health check grace period**, enter the amount of time until Amazon EC2 Auto Scaling checks the Elastic Load Balancing health status of an instance after it enters the **InService** state. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).
 - d. Under **Additional settings, Monitoring**, choose whether to enable CloudWatch group metrics collection. These metrics provide measurements that can be indicators of a potential issue, such as number of terminating instances or number of pending instances. For more information, see [Monitor CloudWatch metrics for your Auto Scaling groups and instances \(p. 260\)](#).
 - e. For **Enable default instance warmup**, select this option and choose the warm-up time for your application. If you are creating an Auto Scaling group that has a scaling policy, the default instance warmup feature improves the Amazon CloudWatch metrics used for dynamic scaling. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).
10. (Optional) On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**:
 - a. For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see [Set capacity limits on your Auto Scaling group \(p. 122\)](#).
 - b. To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see [Target tracking scaling policies for Amazon EC2 Auto Scaling \(p. 135\)](#).
 - c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see [Use instance scale-in protection \(p. 240\)](#).
 11. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).
 12. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see [Tag Auto Scaling groups and instances \(p. 91\)](#).
 13. On the **Review** page, choose **Create Auto Scaling group**.

To create an Auto Scaling group using the command line

You can use one of the following commands:

- [create-auto-scaling-group](#) (AWS CLI)
- [New-ASAutoScalingGroup](#) (AWS Tools for Windows PowerShell)

Create an Auto Scaling group using parameters from an existing instance

Important

This topic only applies to creating an Auto Scaling group using a launch configuration. We provide information about launch configurations for customers who have not yet migrated from launch configurations to launch templates.

If this is your first time creating an Auto Scaling group, we recommend you use the console to create a launch template from an existing EC2 instance. Then use the launch template to create a new Auto Scaling group. For this procedure, see [Create an Auto Scaling group using the Amazon EC2 launch wizard \(p. 89\)](#).

The following procedure shows how to create an Auto Scaling group by specifying an existing instance to use as a base for launching other instances. Multiple parameters are required to create an EC2 instance, such as the Amazon Machine Image (AMI) ID, instance type, key pair, and security group. All of this information is also used by Amazon EC2 Auto Scaling to launch instances on your behalf when there is a need to scale. This information is stored in either a launch template or a launch configuration.

When you use an existing instance, Amazon EC2 Auto Scaling creates an Auto Scaling group that launches instances based on a launch configuration that's created at the same time. The new launch configuration has the same name as the Auto Scaling group, and it includes certain configuration details from the identified instance.

The following configuration details are copied from the identified instance into the launch configuration:

- AMI ID
- Instance type
- Key pair
- Security groups
- IP address type (public or private)
- IAM instance profile, if applicable
- Monitoring (true or false)
- EBS optimized (true or false)
- Tenancy setting, if launching into a VPC (shared or dedicated)
- Kernel ID and RAM disk ID, if applicable
- User data, if specified
- Spot (maximum) price

The following configuration details are not copied from your identified instance:

- Storage: The block devices (EBS volumes and instance store volumes) are not copied from the identified instance. Instead, the block device mapping created as part of creating the AMI determines which devices are used.
- Number of network interfaces: The network interfaces are not copied from your identified instance. Instead, Amazon EC2 Auto Scaling uses its default settings to create one network interface, which is the primary network interface (eth0).
- Instance metadata options: The metadata accessible, metadata version, and token response hop limit settings are not copied from the identified instance. Instead, Amazon EC2 Auto Scaling uses its default settings. For more information, see [Configure the instance metadata options \(p. 42\)](#).
- Load balancers: If the identified instance is registered with one or more load balancers, the information about the load balancer is not copied to the load balancer or target group attribute of the new Auto Scaling group.

- **Tags:** If the identified instance has tags, the tags are not copied to the Tags attribute of the new Auto Scaling group.

The new Auto Scaling group launches instances into the same Availability Zone, VPC, and subnet in which the identified instance is located.

If the identified instance is in a placement group, the new Auto Scaling group launches instances into the same placement group as the identified instance. Because the launch configuration settings do not allow a placement group to be specified, the placement group is copied to the PlacementGroup attribute of the new Auto Scaling group.

Prerequisites

The EC2 instance must meet the following criteria:

- The instance is in the subnet and Availability Zone in which you want to create the Auto Scaling group.
- The instance is not a member of another Auto Scaling group.
- The instance is in the `running` state.
- The AMI that was used to launch the instance must still exist.

Create an Auto Scaling group from an EC2 instance (console)

To create an Auto Scaling group from an EC2 instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Instances**, choose **Instances**, and then select an instance.
3. Choose **Actions**, **Instance settings**, **Attach to Auto Scaling Group**.
4. On the **Attach to Auto Scaling group** page, for **Auto Scaling Group**, enter a name for the group, and then choose **Attach**.

After the instance is attached, it is considered part of the Auto Scaling group. The new Auto Scaling group is created using a new launch configuration with the same name that you specified for the Auto Scaling group. The Auto Scaling group has a desired capacity and maximum size of 1.

5. (Optional) To edit the settings for the Auto Scaling group, on the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**. Select the check box next to the new Auto Scaling group, choose the **Edit** button that is above the list of groups, change the settings as needed, and then choose **Update**.

Create an Auto Scaling group from an EC2 instance (AWS CLI)

In this exercise, we demonstrate how to use the AWS CLI to create an Auto Scaling group from an EC2 instance.

This procedure does not add the instance to the Auto Scaling group. For the instance to be attached, you must run the `attach-instances` command after the Auto Scaling group has been created.

Before you begin, find the ID of the EC2 instance using the Amazon EC2 console or the `describe-instances` command.

To use your current instance as a template

- Use the following [create-auto-scaling-group](#) command to create an Auto Scaling group, `my-asg-from-instance`, from the EC2 instance `i-0e69cc3f05f825f4f`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
--instance-id i-0e69cc3f05f825f4f --min-size 1 --max-size 2 --desired-capacity 2
```

To verify that your Auto Scaling group has launched instances

- Use the following [describe-auto-scaling-groups](#) command to verify that the Auto Scaling group was created successfully.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

The following example response shows that the desired capacity of the group is 2, the group has 2 running instances, and the launch configuration is named `my-asg-from-instance`.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "my-asg-from-instance",
      "AutoScalingGroupARN": "arn",
      "LaunchConfigurationName": "my-asg-from-instance",
      "MinSize": 1,
      "MaxSize": 2,
      "DesiredCapacity": 2,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "us-west-2a"
      ],
      "LoadBalancerNames": [
      ],
      "TargetGroupARNs": [
      ],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 0,
      "Instances": [
        {
          "InstanceId": "i-06905f55584de02da",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "us-west-2a",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "my-asg-from-instance",
          "ProtectedFromScaleIn": false
        },
        {
          "InstanceId": "i-087b42219468eacde",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "us-west-2a",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "my-asg-from-instance",
          "ProtectedFromScaleIn": false
        }
      ],
      "CreatedTime": "2020-10-28T02:39:22.152Z",
      "SuspendedProcesses": [
      ],
      "VPCZoneIdentifier": "subnet-6bea5f06",
    }
  ]
}
```

```
"EnabledMetrics":[ ],
"Tags":[ ],
"TerminationPolicies":[
  "Default"
],
"NewInstancesProtectedFromScaleIn":false,
"ServiceLinkedRoleARN":"arn"
}
]
```

To view the launch configuration

- Use the following [describe-launch-configurations](#) command to view the details of the launch configuration.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-asg-from-instance
```

The following is example output:

```
{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "my-asg-from-instance",
      "LaunchConfigurationARN": "arn",
      "ImageId": "ami-0528a5175983e7f28",
      "KeyName": "my-key-pair-uswest2",
      "SecurityGroups": [
        "sg-05eaec502fcadac2e"
      ],
      "ClassicLinkVPCSecurityGroups": [],
      "UserData": "",
      "InstanceType": "t2.micro",
      "KernelId": "",
      "RamdiskId": "",
      "BlockDeviceMappings": [],
      "InstanceMonitoring": {
        "Enabled": true
      },
      "CreatedTime": "2020-10-28T02:39:22.321Z",
      "EbsOptimized": false,
      "AssociatePublicIpAddress": true
    }
  ]
}
```

To terminate the instance

- If you no longer need the instance, you can terminate it. The following [terminate-instances](#) command terminates the instance *i-0e69cc3f05f825f4f*.

```
aws ec2 terminate-instances --instance-ids i-0e69cc3f05f825f4f
```

After you terminate an Amazon EC2 instance, you can't restart the instance. After termination, its data is gone and the volume can't be attached to any instance. To learn more about terminating instances, see [Terminate an instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Create an Auto Scaling group using the Amazon EC2 launch wizard

The following procedure shows how to create an Auto Scaling group by using the **Launch instance** wizard in the Amazon EC2 console. This option automatically populates a launch template with certain configuration details from the **Launch instance** wizard.

Note

The wizard does not populate the Auto Scaling group with the number of instances you specify; it only populates the launch template with the Amazon Machine Image (AMI) ID and instance type. Use the **Create Auto Scaling group** wizard to specify the number of instances to launch. An AMI provides the information required to configure an instance. You can launch multiple instances from a single AMI when you need multiple instances with the same configuration. We recommend using a custom AMI that already has your application installed on it to avoid having your instances terminated if you reboot an instance belonging to an Auto Scaling group. To use a custom AMI with Amazon EC2 Auto Scaling, you must first create your AMI from a customized instance, and then use the AMI to create a launch template for your Auto Scaling group.

Prerequisites

- You must have created a custom AMI in the same AWS Region where you plan to create the Auto Scaling group. For more information, see [Create an AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.
- You must have IAM permissions to create an Auto Scaling group using a launch template and also to create EC2 resources for the instances. For more information, see [Launch template support \(p. 347\)](#).

To use a custom AMI as a template

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation bar at the top of the screen, the current AWS Region is displayed. Select a Region in which to launch your Auto Scaling group.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch instance**, and then do the following:
 - a. Under **Name and tags**, leave **Name** blank. The name isn't part of the data that's used to create a launch template.
 - b. Under **Application and OS Images (Amazon Machine Image)**, choose **Browse more AMIs** to browse the full AMI catalog.
 - c. Choose **My AMIs**, find the AMI that you created, and then choose **Select**.
 - d. Under **Instance type**, choose an instance type.

Note

Choose the same instance type that you used when you created the AMI or a more powerful one.

- e. On the right side of the screen, under **Summary**, for **Number of instances**, enter any number. The number that you enter here isn't important. You will specify the number of instances that you want to launch when you create the Auto Scaling group.

Under the **Number of instances** field, a message displays that says **When launching more than 1 instance, consider EC2 Auto Scaling**.

- f. Choose the **consider EC2 Auto Scaling** hyperlink text.

- g. On the **Launch into Auto Scaling Group** confirmation dialogue, choose **Continue** to go to the **Create launch template** page with the AMI and instance type you selected in the launch instance wizard already populated.

Create a launch template

After you choose **Continue**, the **Create launch template** page opens. Follow this procedure to create a launch template. For more information, see [Create your launch template \(console\) \(p. 23\)](#).

To create a launch template

1. Under **Launch template name and description**, enter a name and description for the new launch template.
2. (Optional) Under **Key pair (login)**, for **Key pair name**, choose the name of the previously created key pair to use when connecting to instances, for example, using SSH.
3. (Optional) Under **Network settings**, for **Security groups**, choose one or more previously created **security groups**.
4. (Optional) Under **Configure storage**, update the storage configuration. The default storage configuration is determined by the AMI and the instance type.
5. When you are done configuring the launch template, choose **Create launch template**.
6. On the confirmation page, choose **Create Auto Scaling group**.

Create an Auto Scaling group

Note

The rest of this topic describes the basic procedure for creating an Auto Scaling group. For more description of the parameters you can configure for your Auto Scaling group, see [Create an Auto Scaling group using a launch template \(p. 81\)](#).

After you choose **Create Auto Scaling group**, the **Create Auto Scaling group** wizard opens. Follow this procedure to create an Auto Scaling group.

To create an Auto Scaling group

1. On the **Choose launch template or configuration** page, enter a name for the Auto Scaling group.
2. The launch template that you created is already selected for you.

For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

3. Choose **Next** to continue to the next step.
4. On the **Choose instance launch options** page, under **Network**, for **VPC**, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch template.

Tip

If you didn't specify a security group in your launch template, your instances are launched with a default security group from the VPC that you specify. By default, this security group doesn't allow inbound traffic from external networks.

5. For **Availability Zones and subnets**, choose one or more subnets in the specified VPC.
6. Choose **Next** twice to go to the **Configure group size and scaling policies** page.
7. Under **Group size**, define the **Desired capacity** (initial number of instances to launch immediately after the Auto Scaling group is created), **Minimum capacity** (minimum number of instances), and **Maximum capacity** (maximum number of instances).

8. Choose **Skip to review**.
9. On the **Review** page, choose **Create Auto Scaling group**.

Next steps

You can check that the Auto Scaling group has been created correctly by viewing the activity history. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched instances. If the instances fail to launch or they launch but then immediately terminate, see the following topics for possible causes and resolutions:

- [Troubleshoot Amazon EC2 Auto Scaling: EC2 instance launch failures \(p. 362\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: AMI issues \(p. 368\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: Health checks \(p. 372\)](#)

You can now attach a load balancer in the same Region as your Auto Scaling group, if desired. For more information, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#).

Tag Auto Scaling groups and instances

A *tag* is a custom attribute label that you assign or that AWS assigns to an AWS resource. Each tag has two parts:

- A tag key (for example, `costcenter`, `environment`, or `project`)
- An optional field known as a tag value (for example, `111122223333` or `production`)

Tags help you do the following:

- Track your AWS costs. You activate these tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see [Using cost allocation tags](#) in the [AWS Billing User Guide](#).
- Control access to Auto Scaling groups based on tags. You can use conditions in your IAM policies to control access to Auto Scaling groups based on the tags on that group. For more information, see [Tags for security \(p. 95\)](#).
- Filter and search for Auto Scaling groups based on the tags that you add. For more information, see [Use tags to filter Auto Scaling groups \(p. 96\)](#).
- Identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related.

You can tag new or existing Auto Scaling groups. You can also propagate tags from an Auto Scaling group to the EC2 instances that it launches.

Tags are not propagated to Amazon EBS volumes. To add tags to Amazon EBS volumes, specify the tags in a launch template. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

You can create and manage tags through the AWS Management Console, AWS CLI, or SDKs.

Contents

- [Tag naming and usage restrictions \(p. 92\)](#)

- [EC2 instance tagging lifecycle \(p. 92\)](#)
- [Tag your Auto Scaling groups \(p. 92\)](#)
- [Delete tags \(p. 94\)](#)
- [Tags for security \(p. 95\)](#)
- [Control access to tags \(p. 96\)](#)
- [Use tags to filter Auto Scaling groups \(p. 96\)](#)

Tag naming and usage restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 50.
- The maximum number of tags that you can add or remove using a single call is 25.
- The maximum key length is 128 Unicode characters.
- The maximum value length is 256 Unicode characters.
- Tag keys and values are case-sensitive. As a best practice, decide on a strategy for capitalizing tags, and consistently implement that strategy across all resource types.
- Do not use the `aws:` prefix in your tag names or values, because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix, and they do not count toward your tags per resource quota.

EC2 instance tagging lifecycle

If you have opted to propagate tags to your EC2 instances, the tags are managed as follows:

- When an Auto Scaling group launches instances, it adds tags to the instances during resource creation rather than after the resource is created.
- The Auto Scaling group automatically adds a tag to instances with a key of `aws:autoscaling:groupName` and a value of the Auto Scaling group name.
- If you specify instance tags in your launch template and you opted to propagate your group's tags to its instances, all the tags are merged. If the same tag key is specified for a tag in your launch template and a tag in your Auto Scaling group, then the tag value from the group takes precedence.
- When you attach existing instances, the Auto Scaling group adds the tags to the instances, overwriting any existing tags with the same tag key. It also adds a tag with a key of `aws:autoscaling:groupName` and a value of the Auto Scaling group name.
- When you detach an instance from an Auto Scaling group, it removes only the `aws:autoscaling:groupName` tag.

Tag your Auto Scaling groups

When you add a tag to your Auto Scaling group, you can specify whether it should be added to instances launched in the Auto Scaling group. If you modify a tag, the updated version of the tag is added to instances launched in the Auto Scaling group after the change. If you create or modify a tag for an Auto Scaling group, these changes are not made to instances that are already running in the Auto Scaling group.

Contents

- [Add or modify tags \(console\) \(p. 93\)](#)
- [Add or modify tags \(AWS CLI\) \(p. 93\)](#)

Add or modify tags (console)

To tag an Auto Scaling group on creation

When you use the Amazon EC2 console to create an Auto Scaling group, you can specify tag keys and values on the **Add tags** page of the Create Auto Scaling group wizard. To propagate a tag to the instances launched in the Auto Scaling group, make sure that you keep the **Tag new instances** option for that tag selected. Otherwise, you can deselect it.

To add or modify tags for an existing Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Tags, Edit**.
4. To modify existing tags, edit **Key** and **Value**.
5. To add a new tag, choose **Add tag** and edit **Key** and **Value**. You can keep **Tag new instances** selected to add the tag to the instances launched in the Auto Scaling group automatically, and deselect it otherwise.
6. When you have finished adding tags, choose **Update**.

Add or modify tags (AWS CLI)

The following examples show how to use the AWS CLI to add tags when you create Auto Scaling groups, and to add or modify tags for existing Auto Scaling groups.

To tag an Auto Scaling group on creation

Use the `create-auto-scaling-group` command to create a new Auto Scaling group and add a tag, for example, `environment=production`, to the Auto Scaling group. The tag is also added to any instances launched in the Auto Scaling group.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
--launch-configuration-name my-launch-config --min-size 1 --max-size 3 \
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
--tags Key=environment,Value=production,PropagateAtLaunch=true
```

To create or modify tags for an existing Auto Scaling group

Use the `create-or-update-tags` command to create or modify a tag. For example, the following command adds the `Name=my-asg` and `costcenter=cc123` tags. The tags are also added to any instances launched in the Auto Scaling group after this change. If a tag with either key already exists, the existing tag is replaced. The Amazon EC2 console associates the display name for each instance with the name that is specified for the `Name` key (case-sensitive).

```
aws autoscaling create-or-update-tags \
--tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-asg,PropagateAtLaunch=true \
ResourceId=my-asg,ResourceType=auto-scaling-group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

Describe the tags for an Auto Scaling group (AWS CLI)

If you want to view the tags that are applied to a specific Auto Scaling group, you can use either of the following commands:

- [describe-tags](#) — You supply your Auto Scaling group name to view a list of the tags for the specified group.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

The following is an example response.

```
{  
  "Tags": [  
    {  
      "ResourceType": "auto-scaling-group",  
      "ResourceId": "my-asg",  
      "PropagateAtLaunch": true,  
      "Value": "production",  
      "Key": "environment"  
    }  
  ]  
}
```

- [describe-auto-scaling-groups](#) — You supply your Auto Scaling group name to view the attributes of the specified group, including any tags.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupARN": "arn",  
      "HealthCheckGracePeriod": 0,  
      "SuspendedProcesses": [],  
      "DesiredCapacity": 1,  
      "Tags": [  
        {  
          "ResourceType": "auto-scaling-group",  
          "ResourceId": "my-asg",  
          "PropagateAtLaunch": true,  
          "Value": "production",  
          "Key": "environment"  
        }  
      ],  
      "EnabledMetrics": [],  
      "LoadBalancerNames": [],  
      "AutoScalingGroupName": "my-asg",  
      ...  
    }  
  ]  
}
```

Delete tags

You can delete a tag associated with your Auto Scaling group at any time.

Contents

- [Delete tags \(console\) \(p. 95\)](#)

- [Delete tags \(AWS CLI\) \(p. 95\)](#)

Delete tags (console)

To delete a tag

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Tags, Edit**.
4. Choose **Remove** next to the tag.
5. Choose **Update**.

Delete tags (AWS CLI)

Use the `delete-tags` command to delete a tag. For example, the following command deletes a tag with a key of **environment**.

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-group,Key=environment"
```

You must specify the tag key, but you don't have to specify the value. If you specify a value and the value is incorrect, the tag is not deleted.

Tags for security

Use tags to control which Auto Scaling groups can be created, modified, or deleted by IAM users and groups in your account. Provide tag information in the condition element of an IAM policy by using one or more of the following condition keys:

- Use `autoscaling:ResourceTag/tag-key`: *tag-value* to allow (or deny) user actions on Auto Scaling groups with specific tags.
- Use `aws:RequestTag/tag-key`: *tag-value* to require that a specific tag be present (or not present) in a request.
- Use `aws:TagKeys [tag-key, ...]` to require that specific tag keys be present (or not present) in a request.

For example, you could deny access to all Auto Scaling groups that include a tag with the key **environment** and the value **production**, as shown in the following example.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": [  
        "autoscaling:CreateAutoScalingGroup",  
        "autoscaling:UpdateAutoScalingGroup",  
        "autoscaling:DeleteAutoScalingGroup"  
      ]  
    }  
  ]  
}
```

```
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {"autoscaling:ResourceTag/environment": "production"}
        }
    ]
}
```

For more examples, see [Amazon EC2 Auto Scaling identity-based policy examples \(p. 337\)](#).

Control access to tags

IAM also supports controlling which IAM users and groups in your account have permissions to add, modify, or delete tags for Auto Scaling groups.

For example, you could create an IAM policy that allows removing only the tag with the **temporary** key from Auto Scaling groups.

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

For more examples, see [Amazon EC2 Auto Scaling identity-based policy examples \(p. 337\)](#).

Note

Even if you have a policy that restricts your users from performing a tagging (or untagging) operation on an Auto Scaling group, this does not prevent them from manually changing the tags on the instances after they have launched. For examples that control access to tags on EC2 instances, see [Example: Tagging resources](#) in the *Amazon EC2 User Guide for Linux Instances*.

Use tags to filter Auto Scaling groups

The following examples show you how to use filters with the `describe-auto-scaling-groups` command to describe Auto Scaling groups with specific tags. Filtering by tags is limited to the AWS CLI or an SDK, and is not available from the console.

Filtering considerations

- You can specify multiple filters and multiple filter values in a single request.
- You cannot use wildcards with the filter values.
- Filter values are case-sensitive.

Example: Describe Auto Scaling groups with a specific tag key and value pair

The following command shows how to filter results to show only Auto Scaling groups with the tag key and value pair of **environment=production**.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

The following is an example response.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 0,
      "SuspendedProcesses": [],
      "DesiredCapacity": 1,
      "Tags": [
        {
          "ResourceType": "auto-scaling-group",
          "ResourceId": "my-asg",
          "PropagateAtLaunch": true,
          "Value": "production",
          "Key": "environment"
        }
      ],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-asg",
      ...
    }
  ]
}
```

Alternatively, you can specify tags using a tag:<key> filter. For example, the following command shows how to filter results to show only Auto Scaling groups with a tag key and value pair of **environment=production**. This filter is formatted as follows: Name=tag:<key>,Values=<value>, with <key> and <value> representing a tag key and value pair.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag:environment,Values=production
```

You can also filter AWS CLI output by using the --query option. The following example shows how to limit AWS CLI output for the previous command to the group name, minimum size, maximum size, and desired capacity attributes only.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag:environment,Values=production \
--query "AutoScalingGroups[].[AutoScalingGroupName: AutoScalingGroupName, MinSize: MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity]"
```

The following is an example response.

```
[
  {
    "AutoScalingGroupName": "my-asg",
    "MinSize": 0,
    "MaxSize": 10,
    "DesiredCapacity": 1
  }
  ...
]
```

]

For more information about filtering, see [Filtering AWS CLI output](#) in the *AWS Command Line Interface User Guide*.

Example: Describe Auto Scaling groups with tags that match the tag key specified

The following command shows how to filter results to show only Auto Scaling groups with the **environment** tag, regardless of the tag value.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-key,Values=environment
```

Example: Describe Auto Scaling groups with tags that match the set of tag keys specified

The following command shows how to filter results to show only Auto Scaling groups with tags for **environment** and **project**, regardless of the tag values.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

Example: Describe Auto Scaling groups with tags that match at least one of the tag keys specified

The following command shows how to filter results to show only Auto Scaling groups with tags for **environment** or **project**, regardless of the tag values.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-key,Values=environment,project
```

Example: Describe Auto Scaling groups with the specified tag value

The following command shows how to filter results to show only Auto Scaling groups with a tag value of **production**, regardless of the tag key.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-value,Values=production
```

Example: Describe Auto Scaling groups with the set of tag values specified

The following command shows how to filter results to show only Auto Scaling groups with the tag values **production** and **development**, regardless of the tag key.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-value,Values=production Name=tag-value,Values=development
```

Example: Describe Auto Scaling groups with tags that match at least one of the tag values specified

The following command shows how to filter results to show only Auto Scaling groups with a tag value of **production** or **development**, regardless of the tag key.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag-value,Values=production,development
```

Example: Describe Auto Scaling groups with tags that match multiple tag keys and values

You can also combine filters to create custom AND and OR logic to do more complex filtering.

The following command shows how to filter results to show only Auto Scaling groups with a specific set of tags. One tag key is **environment** AND the tag value is (**production** OR **development**) AND the other tag key is **costcenter** AND the tag value is **cc123**.

```
aws autoscaling describe-auto-scaling-groups \
--filters Name=tag:environment,Values=production,development
Name=tag:costcenter,Values=cc123
```

Replace Auto Scaling instances

Amazon EC2 Auto Scaling offers capabilities that let you replace instances after you update or modify your Auto Scaling group. Amazon EC2 Auto Scaling also helps you streamline updates by giving you the option of including them in the same operation that replaces the instances.

This section includes information to help you do the following:

- Start an instance refresh to replace instances in your Auto Scaling group.
- Declare specific updates that describe a desired configuration and update the Auto Scaling group to the desired configuration.
- Skip replacing already updated instances.
- Use checkpoints to replace instances in phases and perform verifications on your instances at specific points.
- Receive notifications by email when a checkpoint is reached.
- Limit the lifetime of instances to ensure consistent software versions and instance configurations across the Auto Scaling group.

Contents

- [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#)
- [Check the status of an instance refresh \(p. 106\)](#)
- [Instance refresh examples that enable skip matching with the AWS Command Line Interface \(AWS CLI\) \(p. 107\)](#)
- [Add checkpoints to an instance refresh \(p. 111\)](#)
- [Replace Auto Scaling instances based on maximum instance lifetime \(p. 114\)](#)

Replace Auto Scaling instances based on an instance refresh

You can use an instance refresh to update the instances in your Auto Scaling group instead of manually replacing instances a few at a time. This can be useful when a configuration change requires you to replace instances, and you have a large number of instances in your Auto Scaling group.

An instance refresh can be helpful when you have a new Amazon Machine Image (AMI) or a new user data script. To use an instance refresh, first create a new launch template that specifies the new AMI or user data script. Then, start an instance refresh to begin updating the instances in the group immediately.

How it works

The following example steps show how an instance refresh works:

- You create a launch template or a new launch template version with your desired updates. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).
- You configure the minimum healthy percentage, instance warmup, and checkpoints, specify your desired configuration that includes your launch template, and start an instance refresh. The desired configuration can optionally specify whether a [mixed instances policy \(p. 52\)](#) is to be applied.
- Amazon EC2 Auto Scaling starts performing a rolling replacement of the instances. It takes a set of instances out of service, terminates them, and launches a set of instances with the new desired configuration. Then, it waits until the instances pass your health checks and complete warmup before it moves on to replacing other instances.
- After a certain percentage of the group is replaced, a checkpoint is reached. Whenever there is a checkpoint, Amazon EC2 Auto Scaling temporarily stops replacing instances, sends a notification, and waits for the amount of time you specified before continuing. After you receive the notification, you can verify that your new instances are working as expected.
- After the instance refresh succeeds, the Auto Scaling group settings are automatically updated with the configuration that you specified at the start of the operation.

Core concepts and terms

Before you get started, familiarize yourself with the following instance refresh core concepts and terms:

Minimum healthy percentage

As part of starting an instance refresh, you specify the minimum healthy percentage to maintain at all times. This is the amount of capacity in an Auto Scaling group that must pass your [health checks \(p. 253\)](#) during an instance refresh so that the operation is allowed to continue. The value is expressed as a percentage of the desired capacity of the Auto Scaling group (rounded up to the nearest integer). Setting the minimum healthy percentage to 100 percent limits the rate of replacement to one instance at a time. In contrast, setting it to 0 percent causes all instances to be replaced at the same time.

Instance warmup

The *instance warmup* is the time period from when a new instance's state changes to `InService` to when it can receive traffic. During an instance refresh, Amazon EC2 Auto Scaling does not immediately move on to the next replacement after determining that a newly launched instance is healthy. It waits for the warm-up period that you specified before it moves on to replacing other instances. This can be helpful when your application takes time to initialize itself before it starts to serve traffic.

Desired configuration

A *desired configuration* is the new configuration you want Amazon EC2 Auto Scaling to deploy across your Auto Scaling group. For example, you can specify the launch template and version for your instances. During an instance refresh, Amazon EC2 Auto Scaling updates the Auto Scaling group to the desired configuration. If a scale-out event occurs during an instance refresh, Amazon EC2 Auto Scaling launches new instances with the desired configuration instead of the group's current settings. After the instance refresh succeeds, Amazon EC2 Auto Scaling updates the settings of the Auto Scaling group to reflect the new desired configuration that you specified as part of the instance refresh.

Skip matching

Skip matching means that Amazon EC2 Auto Scaling skips replacing instances that match the desired configuration. If no desired configuration is specified, then it skips replacing instances that have the same configuration that is already set on the group. If skip matching is not enabled, any instance in the Auto Scaling group can be replaced with a new instance, regardless of whether there are any updates needed.

Checkpoints

A checkpoint is a point in time where the instance refresh pauses for a specified amount of time. An instance refresh can contain multiple checkpoints. Amazon EC2 Auto Scaling emits events for each checkpoint, so that you can add an EventBridge rule to send the events to a target such as Amazon SNS to be notified when a checkpoint is reached. After a checkpoint is reached, you have the opportunity to verify your deployment. If any problems are identified, you can cancel the instance refresh and then roll it back by initiating another instance refresh. The ability to deploy updates in phases is a key benefit of checkpoints. If you don't use checkpoints, rolling replacements are performed continuously.

Considerations

The following are things to consider when starting an instance refresh, to help ensure that the group continues to perform as expected.

- We recommend that you consider enabling the default instance warmup feature to unify warm-up settings at the group level. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).
- While warming up, a newly launched instance is not counted toward the aggregated instance metrics of the Auto Scaling group (such as CPUUtilization, NetworkIn, NetworkOut, and so on).
- If you added scaling policies to the Auto Scaling group, the scaling activities run in parallel. If you set a long interval for the instance refresh warm-up period, it takes more time for newly launched instances to be reflected in the metrics. Therefore, an adequate warm-up period keeps Amazon EC2 Auto Scaling from scaling on stale metric data.
- If you added a lifecycle hook to the Auto Scaling group, the warm-up period does not start until the lifecycle hook actions are complete and the instance enters the InService state. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).
- If you enable skip matching but the launch template, the launch template version, and instance types in the mixed instances policy are not changing, the instance refresh will succeed immediately without making any replacements. If you made any other changes (for example, changing your Spot allocation strategy), Amazon EC2 Auto Scaling updates the settings of the Auto Scaling group to reflect the new desired configuration after the instance refresh succeeds.

Start or cancel an instance refresh (console)

You can create, view, and cancel your instance refreshes with the Amazon EC2 Auto Scaling console. If this is your first time starting an instance refresh, using the console will help you understand the features and options available.

[Start an instance refresh in the console \(basic procedure\)](#)

Use the following procedure if you have not previously defined a [mixed instances policy \(p. 52\)](#) for your Auto Scaling group. If you have previously defined a mixed instances policy, see [Start an instance refresh in the console \(mixed instances group\) \(p. 103\)](#) to start an instance refresh.

To start an instance refresh

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up at the bottom of the **Auto Scaling groups** page.

3. On the **Instance refresh** tab, in **Active instance refresh**, choose **Start instance refresh**.
4. For **Minimum healthy percentage**, enter the percentage of the Auto Scaling group that must remain healthy during an instance refresh. The default is 90 percent. Choosing a lower percentage results in a higher number of instances being terminated and replaced at the same time.
5. For **Instance warmup**, enter the number of seconds from when a new instance's state changes to **InService** to when it can receive traffic, or leave blank to keep the default warmup.

If left blank, the default is the group's default instance warmup if enabled. If default instance warmup is not enabled, then the instance warmup falls back to the value of the health check grace period for the group.

6. (Optional) For **Checkpoints**, choose **Enable checkpoints** to replace instances using an incremental or phased approach to an instance refresh. This provides additional time for verification between sets of replacements. If you choose not to enable checkpoints, the instances are replaced in one nearly continuous operation.

If you enable checkpoints, see [Enable checkpoints \(console\) \(p. 112\)](#) for additional steps.

7. Enable or turn off **Skip matching**:

- To skip replacing instances that already match your current launch template, keep the **Enable skip matching** check box selected.
- If you turn off skip matching by clearing this check box, all instances can be replaced.

When you enable skip matching, instead of using your current launch template, you can set a new launch template or a new version of the current launch template in the **Desired configuration** section of the **Start instance refresh** page.

Note

To use the skip matching feature to update an Auto Scaling group that currently uses a launch configuration, you must select a launch template in **Desired configuration**. Using skip matching with a launch configuration is not supported.

8. (Optional) Expand the **Desired configuration** section to specify any updates you want to make to your Auto Scaling group.

From this point forward, you can choose to use JSON or YAML syntax to edit parameter values instead of making selections in the console interface. To do so, choose **Use code editor** instead of **Use console interface**. The following procedure explains how to make selections using the console interface.

- a. For **Update launch template**:

- If you *haven't* created a new launch template or a new launch template version for your Auto Scaling group, you can keep this check box unselected.
- If you *have* created a new launch template or a new launch template version, select this check box. When you select this option, Amazon EC2 Auto Scaling shows you the current launch template and current launch template version, and it lists any other versions that are available. Choose the launch template and then choose the version.

After you choose a version, you can see the version information. This is the version of the launch template that will be used when replacing instances as part of an instance refresh. If the instance refresh succeeds, this version of the launch template will also be used whenever new instances launch, such as when the group scales.

- b. For **Choose a set of instance types and purchase options to override the instance type in the launch template**:

- Keep this check box unselected to use the instance type and purchase option you specified in your launch template.

- Select this check box if you want to override the instance type in the launch template or take advantage of unused EC2 capacity and save money by running Spot Instances. When you choose this option, you can either manually add each instance type, or choose a primary instance type and a recommendation option that retrieves any additional matching instance types for you. If you plan to launch Spot Instances, we recommend adding a few instance types so that Amazon EC2 Auto Scaling can launch another instance type if there is insufficient instance capacity in your chosen Availability Zones. For further guidance on how to handle these settings, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

Warning

Spot Instances should not be used with applications that are unable to handle a Spot Instance interruption if the Amazon EC2 Spot service needs to reclaim capacity.

If you select this check box, make sure that the launch template doesn't already request Spot Instances. You cannot use a launch template that requests Spot Instances to create an Auto Scaling group that uses multiple instance types and launches Spot and On-Demand Instances.

Note

To configure these options on an Auto Scaling group that currently uses a launch configuration, you must select a launch template in **Update launch template**. Overriding the instance type in your launch configuration is not supported.

9. Review all of your selections to confirm that everything is set up correctly.

At this point, it's a good idea to verify that the differences between the current and proposed changes won't affect your application in unexpected or unwanted ways. For example, if you launched your original instances from a paravirtual (PV) AMI, but want to change to a current generation instance type that is only supported by a hardware virtual machine (HVM) AMI, you need to use an HVM AMI. For more information, see [Compatibility for changing the instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.

10. When you are satisfied with your instance refresh selections, choose **Start**.

[Start an instance refresh in the console \(mixed instances group\)](#)

Use the following procedure if you have created an Auto Scaling group with a [mixed instances policy \(p. 52\)](#). If you haven't defined a mixed instances policy for your group yet, see [Start an instance refresh in the console \(basic procedure\) \(p. 101\)](#) to start an instance refresh.

To start an instance refresh

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up at the bottom of the **Auto Scaling groups** page.

3. On the **Instance refresh** tab, in **Active instance refresh**, choose **Start instance refresh**.
4. For **Minimum healthy percentage**, enter the percentage of the Auto Scaling group that must remain healthy during an instance refresh. The default value is 90 percent. Choosing a lower percentage will result in a higher number of instances being terminated and replaced at the same time.
5. For **Instance warmup**, enter the number of seconds from when a new instance's state changes to **InService** to when it can receive traffic, or leave blank to keep the default warmup.

If left blank, the default is the group's default instance warmup if enabled. If default instance warmup is not enabled, then the instance warmup falls back to the value of the health check grace period for the group.

6. (Optional) For **Checkpoints**, choose **Enable checkpoints** to replace instances using an incremental or phased approach to an instance refresh. This provides additional time for verification between sets of replacements. If you choose not to enable checkpoints, the instances are replaced in one nearly continuous operation.

If you enable checkpoints, see [Enable checkpoints \(console\) \(p. 112\)](#) for additional steps.

7. Enable or turn off **Skip matching**:

- To skip replacing instances that already match your current launch template and any instance type overrides, keep the **Enable skip matching** check box selected.
- If you choose to turn off skip matching by clearing this check box, all instances can be replaced.

When you enable skip matching, instead of using your current launch template, you can set a new launch template or a new version of the current launch template in the **Desired configuration** section of the **Start instance refresh** page. You can also update your instance type overrides in **Desired configuration**.

8. In the **Desired configuration** section, do the following.

From this point forward, you can choose to use JSON or YAML syntax to edit parameter values instead of making selections in the console interface. To do so, choose **Use code editor** instead of **Use console interface**. The following procedure explains how to make selections using the console interface.

- a. For **Update launch template**:

- If you *haven't* created a new launch template or a new launch template version for your Auto Scaling group, you can keep this check box unselected.
- If you have created a new launch template or a new launch template version, select this check box. When you select this option, Amazon EC2 Auto Scaling shows you the current launch template and current launch template version, and it lists any other versions that are available. Choose the launch template and then choose the version.

After you choose a version, you can see the version information. This is the version of the launch template that will be used when replacing instances as part of an instance refresh. If the instance refresh succeeds, this version of the launch template will also be used whenever new instances launch, such as when the group scales.

- b. For **Use these settings to override the instance type and purchase option defined in the launch template**:

By default, this check box is selected. Amazon EC2 Auto Scaling populates each parameter with the value that's currently set in the *mixed instances policy* for the Auto Scaling group. Only update values for the parameters that you want to change. For guidance on these settings, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).

Warning

We recommend that you do not clear this check box. Only clear it if you want to stop using a mixed instances policy. After the instance refresh succeeds, Amazon EC2 Auto Scaling updates your group to match the **Desired configuration**. If it no longer includes a mixed instances policy, Amazon EC2 Auto Scaling gradually terminates any Spot Instances that are currently running and replaces them with On-Demand Instances. Or, if your launch template requests Spot Instances, then Amazon EC2 Auto Scaling gradually terminates any On-Demand Instances that are currently running and replaces them with Spot Instances.

9. Review all of your selections to confirm that everything is set up correctly.

At this point, it's a good idea to verify that the differences between the current and proposed changes won't affect your application in unexpected or unwanted ways. For example, if you

launched your original instances from a paravirtual (PV) AMI, but want to change to a current generation instance type that is only supported by a hardware virtual machine (HVM) AMI, you need to use an HVM AMI. For more information, see [Compatibility for changing the instance type](#) in the [Amazon EC2 User Guide for Linux Instances](#).

When you are satisfied with your instance refresh selections, choose **Start**.

To cancel an instance refresh

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
3. On the **Instance refresh** tab, in **Active instance refresh**, choose **Cancel instance refresh**.
4. When prompted for confirmation, choose **Confirm**.

Start or cancel an instance refresh (AWS CLI)

To start an instance refresh

Use the following `start-instance-refresh` command to start an instance refresh from the AWS CLI. You can specify any preferences that you want to change in a JSON configuration file. When you reference the configuration file, provide the file's path and name as shown in the following example.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json:

```
{  
  "AutoScalingGroupName": "my-asg",  
  "Preferences": {  
    "InstanceWarmup": 60,  
    "MinHealthyPercentage": 50  
  }  
}
```

Alternatively, you can start the instance refresh without the optional preferences by running the following command. If preferences are not provided, the default values are used for `InstanceWarmup` and `MinHealthyPercentage`.

```
aws autoscaling start-instance-refresh --auto-scaling-group-name my-asg
```

Example output:

```
{  
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

Note

To specify your desired configuration and enable skip matching with the AWS CLI, see the additional `start-instance-refresh` examples in [Instance refresh examples that enable skip matching with the AWS Command Line Interface \(AWS CLI\) \(p. 107\)](#).

To cancel an instance refresh

When you cancel an instance refresh using the [cancel-instance-refresh](#) command from the AWS CLI, specify the name of the Auto Scaling group as shown in the following example.

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

Example output:

```
{  
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"  
}
```

Limitations

- **Instances terminated before launch:** When there is only one instance in the Auto Scaling group, starting an instance refresh can result in an outage. This is because Amazon EC2 Auto Scaling terminates an instance and then launches a new instance.
- **Total duration:** The maximum amount of time that an instance refresh can continue to actively replace instances is 14 days.
- **Instances not replaced:** If an instance is on standby or protected from scale in, it cannot be replaced. If Amazon EC2 Auto Scaling encounters an instance that it cannot replace, it will continue to replace other instances.
- **Difference in behavior specific to weighted groups:** If a mixed instances group is configured with an instance weight that is larger than or equal to the group's desired capacity, Amazon EC2 Auto Scaling may replace all InService instances at once. To avoid this situation, follow the recommendation in the [Configure instance weighting for Amazon EC2 Auto Scaling \(p. 68\)](#) topic and specify a desired capacity that is larger than your largest weight when you use weights with your Auto Scaling group.
- **One-hour timeout:** When an instance refresh is unable to continue making replacements because your application doesn't pass health checks or there are instances on standby or protected from scale in, it keeps retrying for an hour. It also provides a status message to help you resolve the issue. If the problem persists after an hour, the operation fails. The intention is to give it time to recover if there is a temporary issue.
- **No rollback:** You can cancel an instance refresh at any time, but any instances that have already been replaced are not rolled back to their previous configuration. If an instance refresh fails, any instances that were already replaced are not rolled back to their previous configuration. To fix a failed instance refresh, first resolve the underlying issue that caused the update to fail, and then initiate another instance refresh.

Check the status of an instance refresh

You can get the status of an instance refresh for your Auto Scaling group using the console or the AWS CLI.

Tip

In the following procedure, you look at the **Instance refresh history**, **Activity history**, and **Instances** sections for the Auto Scaling group. In each, the named columns should already be displayed. To display hidden columns or change the number of rows shown, choose the gear icon on the top right corner of each section to open the preferences modal, update the settings as needed, and choose Confirm.

To check the status of an instance refresh (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Instance refresh** tab, under **Instance refresh history**, you can determine the status of your request by looking at the **Status** column. The operation goes into Pending status while it is initializing. The status should then quickly change to InProgress. When all instances are updated, the status changes to Successful.
4. On the **Activity** tab, under **Activity history**, when the instance refresh starts, you see entries when instances are terminated and another set of entries when instances are launched. In the **Description** column, you can find the instance ID.
5. (Optional) If you have a lot of scaling activities, you can choose the > icon at the top edge of the activity history to see the next page of scaling activities.
6. On the **Instance management** tab, under **Instances**, you can verify that your instances launched successfully. Initially, your instances are in the Pending state. After an instance is ready to receive traffic, its state is InService. The **Health status** column shows the result of the health checks on your instances.

To check the status of an instance refresh (AWS CLI)

View the instance refreshes for an Auto Scaling group by using the following [describe-instance-refreshes](#) command.

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

Example output:

```
{  
  "InstanceRefreshes": [  
    {  
      "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",  
      "AutoScalingGroupName": "my-asg",  
      "Status": "InProgress",  
      "StartTime": "2020-06-02T18:11:27Z",  
      "PercentageComplete": 0,  
      "InstancesToUpdate": 5  
    },  
    {  
      "InstanceRefreshId": "dd7728d0-5bc4-4575-96a3-1b2c52bf8bb1",  
      "AutoScalingGroupName": "my-asg",  
      "Status": "Successful",  
      "StartTime": "2020-06-02T16:43:19Z",  
      "EndTime": "2020-06-02T16:53:37Z",  
      "PercentageComplete": 100,  
      "InstancesToUpdate": 0  
    }  
  ]  
}
```

Instance refresh examples that enable skip matching with the AWS Command Line Interface (AWS CLI)

By default, Amazon EC2 Auto Scaling can replace any instance in an Auto Scaling group during an instance refresh. By enabling skip matching, you can avoid replacing instances that already have your desired configuration.

Skip matching makes it more efficient to:

- Migrate from a launch configuration to the default or latest version of a launch template after launching one or more test instances.
- Migrate from unwanted instance types to instance types that are better suited for your application.
- Roll back changes after one or more instances are replaced as part of a failed or cancelled instance refresh.

Note

The skip matching feature cannot be used to update an Auto Scaling group that uses a launch configuration unless a launch template is specified for the desired configuration.

The following AWS CLI examples demonstrate a few scenarios for the use of skip matching. If you encounter any errors when running these commands, make sure that you have updated the AWS CLI locally to the latest version.

Examples

- [Migrate to the default version of your launch template \(p. 108\)](#)
- [Migrate to the latest version of your launch template \(p. 108\)](#)
- [Skip matching and mixed instances groups \(p. 109\)](#)
 - [Migrate to the default version of your launch template \(p. 109\)](#)
 - [Migrate to the latest version of your launch template \(p. 110\)](#)
 - [Migrate away from unwanted instance types \(p. 110\)](#)

Migrate to the default version of your launch template

The following example shows a [start-instance-refresh](#) command that updates an Auto Scaling group to the default version of your launch template. If there are any instances that are already using the default version of the specified launch template, they are not replaced.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "DesiredConfiguration": {  
        "LaunchTemplate": {  
            "LaunchTemplateId": "lt-068f72b729example",  
            "Version": "$Default"  
        }  
    },  
    "Preferences": {  
        "SkipMatching": true  
    }  
}
```

If successful, this command returns a JSON response that contains the instance refresh ID.

Migrate to the latest version of your launch template

The following example shows a [start-instance-refresh](#) command that updates an Auto Scaling group to the latest version of your launch template. If there are any instances that are already using the latest version of the specified launch template, they are not replaced.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "DesiredConfiguration": {  
        "LaunchTemplate": {  
            "LaunchTemplateId": "lt-068f72b729example",  
            "Version": "$Latest"  
        }  
    },  
    "Preferences": {  
        "SkipMatching": true  
    }  
}
```

If successful, this command returns a JSON response that contains the instance refresh ID.

Skip matching and mixed instances groups

To update a mixed instances group, you must specify settings for a mixed instances policy in your desired configuration. For any mixed instances policy parameters not provided in the desired configuration, Amazon EC2 Auto Scaling resets the parameter value to a default value.

Migrate to the default version of your launch template

The following example shows a [start-instance-refresh](#) command that updates a mixed instances group to the default version of your launch template. If there are any instances that are already using the default version of the specified launch template, they are not replaced.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "DesiredConfiguration": {  
        "MixedInstancesPolicy": {  
            "LaunchTemplate": {  
                "LaunchTemplateSpecification": {  
                    "LaunchTemplateId": "lt-068f72b729example",  
                    "Version": "$Default"  
                },  
                "Overrides": [  
                    ... existing instance types ...  
                ]  
            },  
            "InstancesDistribution": {  
                "OnDemandPercentageAboveBaseCapacity": 50,  
                "SpotAllocationStrategy": "capacity-optimized"  
            }  
        },  
        "Preferences": {  
            "SkipMatching": true  
        }  
    }  
}
```

```
}
```

If successful, this command returns a JSON response that contains the instance refresh ID.

Migrate to the latest version of your launch template

The following example shows a `start-instance-refresh` command that updates a mixed instances group to the latest version of your launch template. If there are any instances that are already using the latest version of the specified launch template, they are not replaced.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json.

```
{
  "AutoScalingGroupName": "my-asg",
  "DesiredConfiguration": {
    "MixedInstancesPolicy": {
      "LaunchTemplate": {
        "LaunchTemplateSpecification": {
          "LaunchTemplateId": "lt-068f72b729example",
          "Version": "$Latest"
        },
        "Overrides": [
          ... existing instance types ...
        ],
        "InstancesDistribution": {
          "OnDemandPercentageAboveBaseCapacity": 50,
          "SpotAllocationStrategy": "capacity-optimized"
        }
      },
      "Preferences": {
        "SkipMatching": true
      }
    }
  }
}
```

If successful, this command returns a JSON response that contains the instance refresh ID.

Migrate away from unwanted instance types

When you have a mixed instances group, you typically have a set of launch template overrides (instance types) that are used to provision instances. The instance types are contained in an `Overrides` section. To tell Amazon EC2 Auto Scaling that you want to replace instances that use a specific instance type, your desired configuration must specify the `Overrides` section without the unwanted instance type. When an instance type in your group doesn't match one of the instance types in the `Overrides` section, the instances are replaced as part of the instance refresh. Note that an instance refresh does not choose the instance pools from which to provision the new instances; instead, the allocation strategies do that.

The following example shows a `start-instance-refresh` command that updates a mixed instances group by replacing any instances that don't match an instance type specified in the desired configuration.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json.

```
{  
  "AutoScalingGroupName": "my-asg",  
  "DesiredConfiguration": {  
    "MixedInstancesPolicy": {  
      "LaunchTemplate": {  
        "LaunchTemplateSpecification": {  
          ... existing launch template and version ...  
        },  
        "Overrides": [  
          {  
            "InstanceType": "c5.large"  
          },  
          {  
            "InstanceType": "c5a.large"  
          },  
          {  
            "InstanceType": "m5.large"  
          },  
          {  
            "InstanceType": "m5a.large"  
          }  
        ]  
      },  
      "InstancesDistribution": {  
        "OnDemandPercentageAboveBaseCapacity": 50,  
        "SpotAllocationStrategy": "capacity-optimized"  
      }  
    },  
    "Preferences": {  
      "SkipMatching": true  
    }  
  }  
}
```

If successful, this command returns a JSON response that contains the instance refresh ID.

Add checkpoints to an instance refresh

When using an instance refresh, you have the option to replace instances in phases, so that you can perform verifications on your instances as you go. To do a phased replacement, you add checkpoints, which are points in time where the instance refresh pauses. Using checkpoints gives you greater control over how you choose to update your Auto Scaling group. It helps you to confirm that your application will function in a reliable, predictable manner.

Amazon EC2 Auto Scaling emits events for each checkpoint. If you add an EventBridge rule to send the events to a target such as Amazon SNS, you can be notified when you can run the required verifications. For more information, see [Create EventBridge rules for instance refresh events \(p. 314\)](#).

Contents

- [Considerations \(p. 111\)](#)
- [Enable checkpoints \(console\) \(p. 112\)](#)
- [Enable checkpoints \(AWS CLI\) \(p. 113\)](#)

Considerations

Keep the following considerations in mind when using checkpoints:

- A checkpoint is reached when the number of instances replaced reaches the percentage threshold that is defined for the checkpoint. The percentage of instances replaced can be equal to or higher than the percentage threshold, but not lower.
- After a checkpoint is reached, the overall percentage complete doesn't display the latest status until after the instances finish warming up.

For example, assume that your Auto Scaling group has 10 instances. Your checkpoint percentages are [20, 50] with a checkpoint delay of 15 minutes and a minimum healthy percentage of 80 percent. Your group makes the following replacements:

- 0:00: Two old instances are replaced with new ones.
- 0:10: Two new instances finish warming up.
- 0:25: Two old instances are replaced with new ones. (To maintain the minimum healthy percentage, only two instances are replaced.)
- 0:35: Two new instances finish warming up.
- 0:35: One old instance is replaced with a new one.
- 0:45: One new instance finishes warming up.

At 0:35, the operation stops launching new instances. The percentage complete doesn't accurately reflect the number of completed replacements yet (50 percent), because the new instance isn't done warming up. After the new instance completes its warm-up period at 0:45, the percentage complete shows 50 percent.

- Because checkpoints are based on percentages, the number of instances to replace changes with the size of the group. When a scale-out activity occurs and the size of the group increases, an in-progress operation could reach a checkpoint again. If that happens, Amazon EC2 Auto Scaling sends another notification and repeats the wait time between checkpoints before continuing.
- It's possible to skip a checkpoint under certain circumstances. For example, suppose that your Auto Scaling group has two instances and your checkpoint percentages are [10, 40, 100]. After the first instance is replaced, Amazon EC2 Auto Scaling calculates that 50 percent of the group was replaced. Because 50 percent is higher than the first two checkpoints, it skips the first checkpoint (10) and sends a notification for the second checkpoint (40).
- Canceling the operation stops any further replacements from being made. If you cancel the operation or it fails before reaching the last checkpoint, any instances that were already replaced are not rolled back to their previous configuration.
- For a partial refresh, when you rerun the operation, Amazon EC2 Auto Scaling doesn't restart from the point of the last checkpoint, nor does it stop when only the old instances are replaced. However, it targets old instances for replacement first, before targeting new instances.

Enable checkpoints (console)

You can enable checkpoints before starting an instance refresh to replace instances using an incremental or phased approach. This provides additional time for verification.

To start an instance refresh that uses checkpoints

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up at the bottom of the **Auto Scaling groups** page.

3. On the **Instance refresh** tab, in **Active instance refresh**, choose **Start instance refresh**.
4. On the **Start instance refresh** page, enter the values for **Minimum healthy percentage** and **Instance warmup**.

5. Select the **Enable checkpoints** check box.

This displays a box where you can define the percentage threshold for the first checkpoint.

6. For **Proceed until ____ % of the group is refreshed**, enter a number (1–100). This sets the percentage for the first checkpoint.
7. To add another checkpoint, choose **Add checkpoint** and then define the percentage for the next checkpoint.
8. To specify how long Amazon EC2 Auto Scaling waits after a checkpoint is reached, update the fields in **Wait for 1 hour between checkpoints**. The time unit can be hours, minutes, or seconds.
9. If you are finished with your instance refresh selections, choose **Start**.

Enable checkpoints (AWS CLI)

To start an instance refresh with checkpoints enabled using the AWS CLI, you need a configuration file that defines the following parameters:

- **CheckpointPercentages**: Specifies threshold values for the percentage of instances to be replaced. These threshold values provide the checkpoints. When the percentage of instances that are replaced and warmed up reaches one of the specified thresholds, the operation waits for a specified period of time. You specify the number of seconds to wait in **CheckpointDelay**. When the specified period of time has passed, the instance refresh continues until it reaches the next checkpoint (if applicable).
- **CheckpointDelay**: Specifies the amount of time, in seconds, to wait after a checkpoint is reached before continuing. Choose a time period that provides enough time to perform your verifications.

The last value shown in the **CheckpointPercentages** array describes the percentage of the Auto Scaling group that needs to be successfully replaced. The operation transitions to **Successful** after this percentage is successfully replaced and each instance is warmed up and ready to serve traffic again.

To create multiple checkpoints

To create multiple checkpoints, use the following example `start-instance-refresh` command. This example configures an instance refresh that initially refreshes one percent of the Auto Scaling group. After waiting 10 minutes, it then refreshes the next 19 percent and waits another 10 minutes. Finally, it refreshes the rest of the group before concluding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json:

```
{  
    "AutoScalingGroupName": "my-asg",  
    "Preferences": {  
        "InstanceWarmup": 60,  
        "MinHealthyPercentage": 80,  
        "CheckpointPercentages": [1, 20, 100],  
        "CheckpointDelay": 600  
    }  
}
```

To create a single checkpoint

To create a single checkpoint, use the following example `start-instance-refresh` command. This example configures an instance refresh that initially refreshes 20 percent of the Auto Scaling group. After waiting 10 minutes, it then refreshes the rest of the group before concluding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json:

```
{  
  "AutoScalingGroupName": "my-asg",  
  "Preferences": {  
    "InstanceWarmup": 60,  
    "MinHealthyPercentage": 80,  
    "CheckpointPercentages": [20, 100],  
    "CheckpointDelay": 600  
  }  
}
```

To partially refresh the Auto Scaling group

To replace only a portion of your Auto Scaling group and then stop completely, use the following example [start-instance-refresh](#) command. This example configures an instance refresh that initially refreshes one percent of the Auto Scaling group. After waiting 10 minutes, it then refreshes the next 19 percent before concluding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of config.json:

```
{  
  "AutoScalingGroupName": "my-asg",  
  "Preferences": {  
    "InstanceWarmup": 60,  
    "MinHealthyPercentage": 80,  
    "CheckpointPercentages": [1, 20],  
    "CheckpointDelay": 600  
  }  
}
```

Replace Auto Scaling instances based on maximum instance lifetime

The maximum instance lifetime specifies the maximum amount of time (in seconds) that an instance can be in service before it is terminated and replaced. A common use case might be a requirement to replace your instances on a schedule because of internal security policies or external compliance controls.

You must specify a value of at least 86,400 seconds (one day). To clear a previously set value, specify a new value of 0. This setting applies to all current and future instances in your Auto Scaling group.

Setting this value too low can cause instances to be replaced faster than desired. In general, Amazon EC2 Auto Scaling replaces instances one at a time, with a pause between replacements. However, if the maximum instance lifetime that you specify doesn't provide enough time to replace each instance individually, Amazon EC2 Auto Scaling must replace more than one instance at a time. Several instances might be replaced at once, by up to 10 percent of the current capacity of your Auto Scaling group.

To manage the rate of replacement, you can do the following:

- Set the maximum instance lifetime limit to a longer period of time. This spaces out the replacements, which is helpful for groups that have a large number of instances to replace.

- Add extra time between certain replacements by using instance protection. This temporarily prevents individual instances in your Auto Scaling group from being replaced. When you're ready to replace these instances, remove instance protection from each individual instance. For more information, see [Use instance scale-in protection \(p. 240\)](#).

Note

Whenever an old instance is replaced and a new instance launches, the new instance uses the launch template or launch configuration that is currently associated with the Auto Scaling group. If your launch template or launch configuration specifies the AMI ID of a different version of your application, this version of your application will be deployed automatically.

Set the maximum instance lifetime

When you create an Auto Scaling group in the console, you cannot set the maximum instance lifetime. However, after the group is created, you can edit it to set the maximum instance lifetime.

To set the maximum instance lifetime for a group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
A split pane opens up at the bottom of the **Auto Scaling groups** page, showing information about the group you selected.
3. On the **Details** tab, choose **Advanced configurations, Edit**.
4. For **Maximum instance lifetime**, enter the maximum number of seconds that an instance can be in service.
5. Choose **Update**.

On the **Activity** tab, under **Activity history**, you can view the replacement of instances in the group throughout its history.

To set the maximum instance lifetime for a group (AWS CLI)

You can also use the AWS CLI to set the maximum instance lifetime for new or existing Auto Scaling groups.

For new Auto Scaling groups, use the [create-auto-scaling-group](#) command.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example config.json file that shows a maximum instance lifetime of 2592000 seconds (30 days).

```
{
  "AutoScalingGroupName": "my-asg",
  "LaunchTemplate": {
    "LaunchTemplateName": "my-launch-template",
    "Version": "$Latest"
  },
  "MinSize": 1,
  "MaxSize": 5,
  "MaxInstanceLifetime": 2592000,
  "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags": []
}
```

}

For existing Auto Scaling groups, use the [update-auto-scaling-group](#) command.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --max-instance-lifetime 2592000
```

To verify the maximum instance lifetime for an Auto Scaling group

Use the [describe-auto-scaling-groups](#) command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
    "AutoScalingGroups": [  
        {  
            "AutoScalingGroupName": "my-asg",  
            "AutoScalingGroupARN": "arn",  
            "LaunchTemplate": {  
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "$Latest"  
            },  
            "MinSize": 1,  
            "MaxSize": 5,  
            "DesiredCapacity": 1,  
            "DefaultCooldown": 300,  
            "AvailabilityZones": [  
                "us-west-2a",  
                "us-west-2b",  
                "us-west-2c"  
            ],  
            "LoadBalancerNames": [],  
            "TargetGroupARNs": [],  
            "HealthCheckType": "EC2",  
            "HealthCheckGracePeriod": 0,  
            "Instances": [  
                {  
                    "InstanceId": "i-04d180b9d5fc578fc",  
                    "InstanceType": "t2.small",  
                    "AvailabilityZone": "us-west-2b",  
                    "LifecycleState": "Pending",  
                    "HealthStatus": "Healthy",  
                    "LaunchTemplate": {  
                        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",  
                        "LaunchTemplateName": "my-launch-template",  
                        "Version": "7"  
                    },  
                    "ProtectedFromScaleIn": false  
                }  
            ],  
            "CreatedTime": "2019-11-14T22:56:15.487Z",  
            "SuspendedProcesses": [],  
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",  
            "EnabledMetrics": [],  
            "Tags": [],  
            "TerminationPolicies": [  
                "Default"  
            ],  
            "NewInstancesProtectedFromScaleIn": false,  
            "HealthCheckGracePeriod": 0  
        }  
    ]  
}
```

```
        "ServiceLinkedRoleARN": "arn",
        "MaxInstanceLifetime": 2592000
    ]
}
```

Limitations

- **Maximum lifetime not guaranteed to be exact for every instance:** Instances are not guaranteed to be replaced only at the end of their maximum duration. In some situations, Amazon EC2 Auto Scaling might need to start replacing instances immediately after you update the maximum instance lifetime parameter. The reason for this behavior is to avoid replacing all instances at the same time.
- **Instances terminated before launch:** When there is only one instance in the Auto Scaling group, the maximum instance lifetime feature can result in an outage because Amazon EC2 Auto Scaling terminates an instance and then launches a new instance.

Delete your Auto Scaling infrastructure

To completely delete your scaling infrastructure, complete the following tasks.

Tasks

- [Delete your Auto Scaling group \(p. 117\)](#)
- [\(Optional\) Delete the launch configuration \(p. 118\)](#)
- [\(Optional\) Delete the launch template \(p. 118\)](#)
- [\(Optional\) Delete the load balancer and target groups \(p. 119\)](#)
- [\(Optional\) Delete CloudWatch alarms \(p. 119\)](#)

Delete your Auto Scaling group

When you delete an Auto Scaling group, its desired, minimum, and maximum values are set to 0. As a result, the instances are terminated. Deleting an instance also deletes any associated logs or data, and any volumes on the instance. If you do not want to terminate one or more instances, you can detach them before you delete the Auto Scaling group. If the group has scaling policies, deleting the group deletes the policies, the underlying alarm actions, and any alarm that no longer has an associated action.

To delete your Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group and choose **Delete**.
3. When prompted for confirmation, type **delete** to confirm deleting the specified Auto Scaling group and then choose **Delete**.

A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. The **Desired**, **Min**, and **Max** columns show 0 instances for the Auto Scaling group. It takes a few minutes to terminate the instance and delete the group. Refresh the list to see the current state.

To delete your Auto Scaling group (AWS CLI)

Use the following [delete-auto-scaling-group](#) command to delete the Auto Scaling group. This operation does not work if the group has any EC2 instances; it is for group's with zero instances only.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

If the group has instances or scaling activities in progress, use the [delete-auto-scaling-group](#) command with the `--force-delete` option. This will also terminate the EC2 instances. When you delete an Auto Scaling group from the Amazon EC2 Auto Scaling console, the console uses this operation to terminate any EC2 instances and delete the group at the same time.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-delete
```

(Optional) Delete the launch configuration

You can skip this step to keep the launch configuration for future use.

To delete the launch configuration (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
3. On the **Launch configurations** page, choose your launch configuration and choose **Actions, Delete launch configuration**.
4. When prompted for confirmation, choose **Delete**.

To delete the launch configuration (AWS CLI)

Use the following [delete-launch-configuration](#) command.

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-config
```

(Optional) Delete the launch template

You can delete your launch template or just one version of your launch template. When you delete a launch template, all its versions are deleted.

You can skip this step to keep the launch template for future use.

To delete your launch template (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Instances**, choose **Launch Templates**.
3. Select your launch template and then do one of the following:
 - Choose **Actions, Delete template**. When prompted for confirmation, type **Delete** to confirm deleting the specified launch template and then choose **Delete**.
 - Choose **Actions, Delete template version**. Select the version to delete and choose **Delete**.

To delete the launch template (AWS CLI)

Use the following [delete-launch-template](#) command to delete your template and all its versions.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

Alternatively, you can use the [delete-launch-template-versions](#) command to delete a specific version of a launch template.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --  
versions 1
```

(Optional) Delete the load balancer and target groups

Skip this step if your Auto Scaling group is not associated with an Elastic Load Balancing load balancer, or if you want to keep the load balancer for future use.

To delete your load balancer (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Choose the load balancer and choose **Actions, Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

To delete your target group (console)

1. On the navigation pane, under **Load Balancing**, choose **Target Groups**.
2. Choose the target group and choose **Actions, Delete**.
3. When prompted for confirmation, choose **Yes, Delete**.

To delete the load balancer associated with the Auto Scaling group (AWS CLI)

For Application Load Balancers and Network Load Balancers, use the following [delete-load-balancer](#) and [delete-target-group](#) commands.

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn  
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

For Classic Load Balancers, use the following [delete-load-balancer](#) command.

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

(Optional) Delete CloudWatch alarms

To delete any CloudWatch alarms associated with your Auto Scaling group, complete the following steps.

You can skip this step if your Auto Scaling group is not associated with any CloudWatch alarms, or if you want to keep the alarms for future use.

Note

Deleting an Auto Scaling group automatically deletes the CloudWatch alarms that Amazon EC2 Auto Scaling manages for a target tracking scaling policy.

To delete the CloudWatch alarms (console)

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. On the navigation pane, choose **Alarms**.
3. Choose the alarms and choose **Action, Delete**.
4. When prompted for confirmation, choose **Delete**.

To delete the CloudWatch alarms (AWS CLI)

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-AddCapacity and Step-Scaling-AlarmLow-RemoveCapacity alarms.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

Scale the size of your Auto Scaling group

Scaling is the ability to increase or decrease the compute capacity of your application. Scaling starts with an event, or scaling action, which instructs an Auto Scaling group to either launch or terminate Amazon EC2 instances.

Amazon EC2 Auto Scaling provides a number of ways to adjust scaling to best meet the needs of your applications. As a result, it's important that you have a good understanding of your application. Keep the following considerations in mind:

- What role should Amazon EC2 Auto Scaling play in your application's architecture? It's common to think about automatic scaling primarily as a way to increase and decrease capacity, but it's also useful for maintaining a steady number of servers.
- What cost constraints are important to you? Because Amazon EC2 Auto Scaling uses EC2 instances, you only pay for the resources that you use. Knowing your cost constraints helps you decide when to scale your applications, and by how much.
- What metrics are important to your application? Amazon CloudWatch supports a number of different metrics that you can use with your Auto Scaling group.

Contents

- [Scaling options \(p. 121\)](#)
- [Set capacity limits on your Auto Scaling group \(p. 122\)](#)
- [Maintain a fixed number of instances in your Auto Scaling group \(p. 123\)](#)
- [Manual scaling for Amazon EC2 Auto Scaling \(p. 124\)](#)
- [Dynamic scaling for Amazon EC2 Auto Scaling \(p. 133\)](#)
- [Predictive scaling for Amazon EC2 Auto Scaling \(p. 168\)](#)
- [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#)
- [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#)
- [Warm pools for Amazon EC2 Auto Scaling \(p. 219\)](#)
- [Control which Auto Scaling instances terminate during scale in \(p. 230\)](#)
- [Temporarily remove instances from your Auto Scaling group \(p. 242\)](#)
- [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#)

Scaling options

Amazon EC2 Auto Scaling provides several ways for you to scale your Auto Scaling group.

Maintain current instance levels at all times

You can configure your Auto Scaling group to maintain a specified number of running instances at all times. To maintain the current instance levels, Amazon EC2 Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When Amazon EC2 Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one. For more information, see [Maintain a fixed number of instances in your Auto Scaling group \(p. 123\)](#).

Scale manually

Manual scaling is the most basic way to scale your resources, where you specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Amazon EC2 Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity. For more information, see [Manual scaling for Amazon EC2 Auto Scaling \(p. 124\)](#).

Scale based on a schedule

Scaling by schedule means that scaling actions are performed automatically as a function of time and date. This is useful when you know exactly when to increase or decrease the number of instances in your group, simply because the need arises on a predictable schedule. For more information, see [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#).

Scale based on demand

A more advanced way to scale your resources, using dynamic scaling, lets you define a scaling policy that dynamically resizes your Auto Scaling group to meet changes in demand. For example, let's say that you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This method is useful for scaling in response to changing conditions, when you don't know when those conditions will change. You can set up Amazon EC2 Auto Scaling to respond for you. For more information, see [Dynamic scaling for Amazon EC2 Auto Scaling \(p. 133\)](#).

Use predictive scaling

You can also combine predictive scaling and dynamic scaling (proactive and reactive approaches, respectively) to scale your EC2 capacity faster. Use predictive scaling to increase the number of EC2 instances in your Auto Scaling group in advance of daily and weekly patterns in traffic flows. For more information, see [Predictive scaling for Amazon EC2 Auto Scaling \(p. 168\)](#).

Set capacity limits on your Auto Scaling group

Capacity limits represent the minimum and maximum group size that you want for your Auto Scaling group. You set limits separately for the minimum and maximum size.

The group's desired capacity can be resized to a number that's within the range of your minimum and maximum size limits. The desired capacity must be equal to or greater than the minimum group size, and equal to or less than the maximum group size.

- **Desired capacity:** Represents the initial capacity of the Auto Scaling group at the time of creation. An Auto Scaling group attempts to maintain the desired capacity. It starts by launching the number of instances that are specified for the desired capacity, and maintains this number of instances as long as there are no scaling policies or scheduled actions attached to the Auto Scaling group.
- **Minimum capacity:** Represents the minimum group size. When scaling policies are set, an Auto Scaling group cannot decrease its desired capacity lower than the minimum size limit.
- **Maximum capacity:** Represents the maximum group size. When scaling policies are set, an Auto Scaling group cannot increase its desired capacity higher than the maximum size limit.

The minimum and maximum size limits also apply in the following scenarios:

- When you manually scale your Auto Scaling group by updating its desired capacity.
- When scheduled actions run that update the desired capacity. If a scheduled action runs without specifying new minimum and maximum size limits for the group, then the group's current minimum and maximum size limits apply.

An Auto Scaling group always tries to maintain its desired capacity. In cases where an instance terminates unexpectedly (for example, because of a Spot Instance interruption, a health check failure, or human action), the group automatically launches a new instance to maintain its desired capacity.

To manage these settings in the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. On the **Auto Scaling groups** page, select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the page.

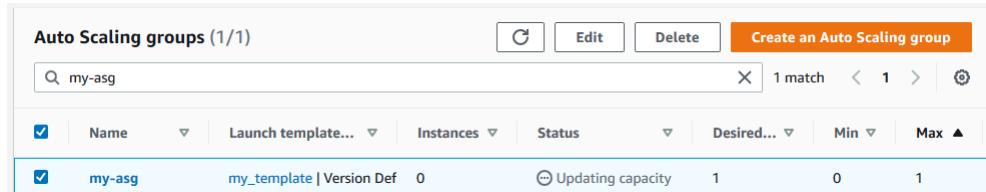
4. In the lower pane, in the **Details** tab, view or change the current settings for minimum, maximum, and desired capacity. For more information, see [Change the size of your Auto Scaling group \(console\) \(p. 124\)](#).

Above the **Details** pane, you can find information such as the current number of instances in the Auto Scaling group, the minimum, maximum, and desired capacity, and a status column. If the Auto Scaling group uses instance weighting, you can also find the number of capacity units contributed to the desired capacity.

To add or remove columns from the list, choose the settings icon at the top of the page. Then, for **Auto Scaling groups attributes**, turn each column on or off, and choose **Confirm**.

To verify the size of your Auto Scaling group after making changes

The **Instances** column shows the number of instances that are currently running. While an instance is being launched or terminated, the **Status** column displays a status of *Updating capacity*, as shown in the following image.



Auto Scaling groups (1/1)							
		Edit		Delete		Create an Auto Scaling group	
<input type="text"/> my-asg		1 match		< 1 >		<input type="button"/>	
	Name	Launch template...	Instances	Status	Desired...	Min	Max
<input checked="" type="checkbox"/>	my-asg	my_template Version Def	0	Updating capacity	1	0	1

Wait for a few minutes, and then refresh the view to see the latest status. After a scaling activity completes, the **Instances** column shows an updated value.

You can view the number of instances and the status of the currently running instances from the **Instance management** tab, under **Instances**.

Note

You can use Service Quotas to update the total capacity limits for EC2 instances and other resources in your AWS account. In the Service Quotas console, you can view all your available service quotas and request increases for them. For more information, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

Maintain a fixed number of instances in your Auto Scaling group

Amazon EC2 Auto Scaling lets you set up an Auto Scaling group to maintain a fixed size. You can then choose whether to adjust the desired capacity of the group or manually add or remove Amazon EC2 instances from the group to handle traffic changes to your application.

If a fixed number of instances is needed, this can be achieved by setting the same value for minimum, maximum, and desired capacity. After you have created your Auto Scaling group, the group starts by launching enough instances to meet its desired capacity. If there are no other scaling conditions attached to the Auto Scaling group, the group maintains this number of running instances at all times.

Your Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. Amazon EC2 Auto Scaling monitors the health of each Auto Scaling instance. When it finds that an instance has become unhealthy, it terminates that instance and launches a new one. Instances can fail a health check because of a variety of reasons. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).

Manual scaling for Amazon EC2 Auto Scaling

At any time, you can change the size of an existing Auto Scaling group manually. You can either update the desired capacity of the Auto Scaling group, or update the instances that are attached to the Auto Scaling group. Manually scaling your group can be useful when automatic scaling is not needed or when you need to hold capacity at a fixed number of instances.

Change the size of your Auto Scaling group (console)

When you change the desired capacity of your Auto Scaling group, Amazon EC2 Auto Scaling manages the process of launching or terminating instances to maintain the new group size.

The following example assumes that you've created an Auto Scaling group with a minimum size of 1 and a maximum size of 5. Therefore, the group currently has one running instance.

To change the size of your Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Details** tab, choose **Group details, Edit**.
4. For **Desired capacity**, increase the desired capacity by one. For example, if the current value is 1, enter 2.

The desired capacity must be less than or equal to the maximum size of the group. If your new value for **Desired capacity** is greater than **Maximum capacity**, you must update **Maximum capacity**.

5. When you are finished, choose **Update**.

Now, verify that your Auto Scaling group has launched one additional instance.

To verify that the size of your Auto Scaling group has changed

1. On the **Activity** tab, in **Activity history**, the **Status** column shows the current status of your instance. Use the refresh button until you see the status of your instance change to **Successful**. This indicates that your Auto Scaling group has successfully launched a new instance.

Note

If the instance fails to launch, you can find troubleshooting tips in [Troubleshoot Amazon EC2 Auto Scaling \(p. 360\)](#).

2. On the **Instance management** tab, in **Instances**, the **Lifecycle** column shows the state of your instances. It takes a short time for an instance to launch. After the instance starts, its state changes

to InService. You can see that your Auto Scaling group has launched 1 new instance, and it is in the InService state.

Change the size of your Auto Scaling group (AWS CLI)

When you change the size of your Auto Scaling group, Amazon EC2 Auto Scaling manages the process of launching or terminating instances to maintain the new group size. The default behavior is not to wait for the default cooldown period to complete, but you can override the default and wait for the cooldown period to complete. For more information, see [Scaling cooldowns for Amazon EC2 Auto Scaling \(p. 153\)](#).

The following example assumes that you've created an Auto Scaling group with a minimum size of 1 and a maximum size of 5. Therefore, the group currently has one running instance.

To change the size of your Auto Scaling group

Use the [set-desired-capacity](#) command to change the size of your Auto Scaling group, as shown in the following example.

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
--desired-capacity 2
```

If you choose to honor the default cooldown period for your Auto Scaling group, you must specify the `--honor-cooldown` option as shown in the following example.

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
--desired-capacity 2 --honor-cooldown
```

To verify the size of your Auto Scaling group

Use the [describe-auto-scaling-groups](#) command to confirm that the size of your Auto Scaling group has changed, as in the following example.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is example output, with details about the group and instances launched.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupARN": "arn:",  
      "ServiceLinkedRoleARN": "arn:",  
      "TargetGroupARNs": [],  
      "SuspendedProcesses": [],  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      },  
      "Tags": [],  
      "EnabledMetrics": [],  
      "LoadBalancerNames": [],  
      "AutoScalingGroupName": "my-asg",  
      "DefaultCooldown": 300,  
      "MinSize": 1,  
      "Instances": [  
        {  
          "ProtectedFromScaleIn": false,  
          "AvailabilityZone": "us-west-2a",  
          "LaunchTemplate": {  
            "LaunchTemplateName": "my-launch-template",  
            "Version": "1",  
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
          },  
          "HealthCheckGracePeriod": 60,  
          "HealthCheckType": "EC2",  
          "InstanceId": "i-000000000000000000",  
          "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCH",  
          "ProtectedFromScaleIn": false,  
          "ProtectedFromScaleOut": false,  
          "Reason": "Launch",  
          "ReasonCode": "LAUNCH",  
          "Status": "InService",  
          "StatusReason": "Launch successful",  
          "StatusReasonCode": "LAUNCH_SUCCESSFUL",  
          "Type": "ON_DEMAND",  
          "UpdateGroup": "my-asg",  
          "UpdateGroupARN": "arn:",  
          "UpdateGroupType": "STANDARD",  
          "UpdateGroupVersion": 1  
        }  
      ]  
    }  
  ]  
}
```

```
"LaunchTemplate": {  
    "LaunchTemplateName": "my-launch-template",  
    "Version": "1",  
    "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
},  
    "InstanceId": "i-05b4f7d5be44822a6",  
    "HealthStatus": "Healthy",  
    "LifecycleState": "Pending"  
},  
{  
    "ProtectedFromScaleIn": false,  
    "AvailabilityZone": "us-west-2a",  
    "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
    },  
    "InstanceId": "i-0c20ac468fa3049e8",  
    "HealthStatus": "Healthy",  
    "LifecycleState": "InService"  
}  
],  
"MaxSize": 5,  
"VPCZoneIdentifier": "subnet-c87f2be0",  
"HealthCheckGracePeriod": 300,  
"TerminationPolicies": [  
    "Default"  
],  
"CreatedTime": "2019-03-18T23:30:42.611Z",  
"AvailabilityZones": [  
    "us-west-2a"  
],  
"HealthCheckType": "EC2",  
"NewInstancesProtectedFromScaleIn": false,  
"DesiredCapacity": 2  
}  
]  
}
```

Notice that DesiredCapacity shows the new value. Your Auto Scaling group has launched an additional instance.

Attach EC2 instances to your Auto Scaling group

Amazon EC2 Auto Scaling provides you with the option of attaching one or more EC2 instances to your existing Auto Scaling group. After an instance is attached, it is considered part of the Auto Scaling group.

When you attach instances, consider the following:

- When you attach instances, the desired capacity of the group increases by the number of instances being attached. If the number of instances being attached plus the desired capacity exceeds the maximum size of the group, the request fails.
- If you attach an instance to an Auto Scaling group that has an attached load balancer target group or Classic Load Balancer, the instance is registered with the load balancer.

For an instance to be attached, it must meet the following criteria:

- The instance is in the `running` state with Amazon EC2.
- The AMI used to launch the instance must still exist.
- The instance is not a member of another Auto Scaling group.

- The instance is launched into one of the Availability Zones defined in your Auto Scaling group.
- If the Auto Scaling group has an attached load balancer target group or Classic Load Balancer, the instance and the load balancer must both be in the same VPC.

Attach an instance (console)

Use the following procedure to attach an instance to your Auto Scaling group.

To attach an instance to an existing Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. (Optional) On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**. Select the Auto Scaling group and verify that the maximum size of the Auto Scaling group is large enough that you can add another instance. Otherwise, on the **Details** tab, increase the maximum capacity.
3. On the navigation pane, under **Instances**, choose **Instances**, and then select an instance.
4. Choose **Actions**, **Instance settings**, **Attach to Auto Scaling Group**.
5. On the **Attach to Auto Scaling group** page, for **Auto Scaling Group**, select the Auto Scaling group, and then choose **Attach**.
6. If the instance doesn't meet the criteria, you get an error message with the details. For example, the instance might not be in the same Availability Zone as the Auto Scaling group. Choose **Close** and try again with an instance that meets the criteria.

Attach an instance (AWS CLI)

Use the following procedure to attach an instance to your Auto Scaling group.

The examples use an Auto Scaling group with the following configuration:

- Auto Scaling group name = my-asg
- Minimum size = 1
- Maximum size = 5
- Desired capacity = 2

To attach an instance to an Auto Scaling group

1. Describe a specific Auto Scaling group using the following `describe-auto-scaling-groups` command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

The following example response shows that the desired capacity is 2 and that the group has two running instances.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupARN": "arn",  
      "ServiceLinkedRoleARN": "arn",  
      "TargetGroupARNs": [],  
      "SuspendedProcesses": [],  
      "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
      }  
    }  
  ]  
}
```

```

},
"Tags": [],
"EnabledMetrics": [],
"LoadBalancerNames": [],
"AutoScalingGroupName": "my-asg",
"DefaultCooldown": 300,
"MinSize": 1,
"Instances": [
{
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-05b4f7d5be44822a6",
    "HealthStatus": "Healthy",
    "LifecycleState": "Pending"
},
{
    "ProtectedFromScaleIn": false,
    "AvailabilityZone": "us-west-2a",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-0c20ac468fa3049e8",
    "HealthStatus": "Healthy",
    "LifecycleState": "InService"
}
],
"MaxSize": 5,
"VPCZoneIdentifier": "subnet-c87f2be0",
"HealthCheckGracePeriod": 300,
"TerminationPolicies": [
    "Default"
],
"CreatedTime": "2019-03-18T23:30:42.611Z",
"AvailabilityZones": [
    "us-west-2a"
],
"HealthCheckType": "EC2",
"NewInstancesProtectedFromScaleIn": false,
"DesiredCapacity": 2
}
]
}
]
}

```

2. Attach an instance to the Auto Scaling group using the following [attach-instances](#) command.

```
aws autoscaling attach-instances --instance-ids i-0787762faf1c28619 --auto-scaling-group-name my-asg
```

3. To verify that the instance is attached, use the following [describe-auto-scaling-groups](#) command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

The following example response shows that the desired capacity has increased by 1 instance (to a new capacity of 3), and that there is a new instance, i-0787762faf1c28619.

```
{
```

```
"AutoScalingGroups": [
  {
    "AutoScalingGroupARN": "arn",
    "ServiceLinkedRoleARN": "arn",
    "TargetGroupARNs": [],
    "SuspendedProcesses": [],
    "LaunchTemplate": {
      "LaunchTemplateName": "my-launch-template",
      "Version": "1",
      "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "Tags": [],
    "EnabledMetrics": [],
    "LoadBalancerNames": [],
    "AutoScalingGroupName": "my-asg",
    "DefaultCooldown": 300,
    "MinSize": 1,
    "Instances": [
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-05b4f7d5be44822a6",
        "HealthStatus": "Healthy",
        "LifecycleState": "Pending"
      },
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0c20ac468fa3049e8",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
      },
      {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
          "LaunchTemplateName": "my-launch-template",
          "Version": "1",
          "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0787762faf1c28619",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService"
      }
    ],
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-c87f2be0",
    "HealthCheckGracePeriod": 300,
    "TerminationPolicies": [
      "Default"
    ],
    "CreatedTime": "2019-03-18T23:30:42.611Z",
    "AvailabilityZones": [
      "us-west-2a"
    ],
    "HealthCheckType": "EC2",
  }
]
```

```
        "NewInstancesProtectedFromScaleIn": false,  
        "DesiredCapacity": 3  
    }  
}
```

Detach EC2 instances from your Auto Scaling group

You can remove (detach) an instance that is in the `InService` state from an Auto Scaling group. After the instance is detached, you can manage it independently from the rest of the Auto Scaling group. By detaching an instance, you can:

- Move an instance out of one Auto Scaling group and attach it to a different group. For more information, see [Attach EC2 instances to your Auto Scaling group \(p. 126\)](#).
- Test an Auto Scaling group by creating it using existing instances running your application. You can then detach these instances from the Auto Scaling group when your tests are complete.

When you detach instances, consider the following:

- If the number of instances that you are detaching decreases the size of the Auto Scaling group below its minimum capacity, you must decrement the minimum capacity for the group before you can detach the instances.
- When you detach instances, you have the option of decrementing the desired capacity for the Auto Scaling group by the number of instances that you are detaching. If you choose not to decrement the capacity, Amazon EC2 Auto Scaling launches new instances to replace the ones that you detach. If you decrement the capacity but detach multiple instances from the same Availability Zone, Amazon EC2 Auto Scaling can rebalance the Availability Zones unless you suspend the AZRebalance process. For more information, see [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#).
- If you detach an instance from an Auto Scaling group that has an attached load balancer target group or Classic Load Balancer, the instance is deregistered from the load balancer. If connection draining (deregistration delay) is enabled for your load balancer, Amazon EC2 Auto Scaling waits for in-flight requests to complete.

Note

If you are detaching instances that are in the `Standby` state, exercise caution. Attempting to detach instances after putting them into the `Standby` state may cause other instances to terminate unexpectedly.

Detach instances (console)

Use the following procedure to detach an instance from your Auto Scaling group.

To detach an instance from an existing Auto Scaling group

- Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
- Select the check box next to your Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
- On the **Instance management** tab, in **Instances**, select an instance and choose **Actions, Detach**.
- In the **Detach instance** dialog box, select the check box to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Choose **Detach instance**.

Detach instances (AWS CLI)

Use the following procedure to detach an instance from your Auto Scaling group.

The examples use an Auto Scaling group with the following configuration:

- Auto Scaling group name = my-asg
- Minimum size = 1
- Maximum size = 5
- Desired capacity = 4

To detach an instance from an existing Auto Scaling group

1. List the current instances using the following [describe-auto-scaling-instances](#) command.

```
aws autoscaling describe-auto-scaling-instances
```

The following example response shows that the group has four running instances.

```
{  
    "AutoScalingInstances": [  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-05b4f7d5be44822a6",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-0c20ac468fa3049e8",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-0787762faf1c28619",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-0787762faf1c28619",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        }  
    ]  
}
```

```
{  
    "ProtectedFromScaleIn": false,  
    "AvailabilityZone": "us-west-2a",  
    "LaunchTemplate": {  
        "LaunchTemplateName": "my-launch-template",  
        "Version": "1",  
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
    },  
    "InstanceId": "i-0f280a4c58d319a8a",  
    "AutoScalingGroupName": "my-asg",  
    "HealthStatus": "HEALTHY",  
    "LifecycleState": "InService"  
}  
]  
}
```

2. Detach an instance and decrement the desired capacity using the following [detach-instances](#) command.

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \  
    --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

3. Verify that the instance is detached using the following [describe-auto-scaling-instances](#) command.

```
aws autoscaling describe-auto-scaling-instances
```

The following example response shows that there are now three running instances.

```
{  
    "AutoScalingInstances": [  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-0c20ac468fa3049e8",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-0787762faf1c28619",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            }  
        }  
    ]  
}
```

```
        },
        "InstanceId": "i-0f280a4c58d319a8a",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
    }
}
```

Dynamic scaling for Amazon EC2 Auto Scaling

Dynamic scaling scales the capacity of your Auto Scaling group as traffic changes occur.

Amazon EC2 Auto Scaling supports the following types of dynamic scaling policies:

- **Target tracking scaling**—Increase and decrease the current capacity of the group based on a Amazon CloudWatch metric and a target value. It works similar to the way that your thermostat maintains the temperature of your home—you select a temperature and the thermostat does the rest.
- **Step scaling**—Increase and decrease the current capacity of the group based on a set of scaling adjustments, known as *step adjustments*, that vary based on the size of the alarm breach.
- **Simple scaling**—Increase and decrease the current capacity of the group based on a single scaling adjustment, with a cooldown period between each scaling activity.

If you are scaling based on a metric that increases or decreases proportionally to the number of instances in an Auto Scaling group, we recommend that you use target tracking scaling policies. Otherwise, we recommend that you use step scaling policies.

With target tracking, an Auto Scaling group scales in direct proportion to the actual load on your application. That means that in addition to meeting the immediate need for capacity in response to load changes, a target tracking policy can also adapt to load changes that take place over time, for example, due to seasonal variations.

By default, new Auto Scaling groups start without any scaling policies. When you use an Auto Scaling group without any form of dynamic scaling, it doesn't scale on its own unless you set up scheduled scaling or predictive scaling.

Contents

- [How dynamic scaling policies work \(p. 133\)](#)
- [Multiple dynamic scaling policies \(p. 134\)](#)
- [Target tracking scaling policies for Amazon EC2 Auto Scaling \(p. 135\)](#)
- [Step and simple scaling policies for Amazon EC2 Auto Scaling \(p. 141\)](#)
- [Set default values for instance warmup or scaling cooldown \(p. 150\)](#)
- [Scaling based on Amazon SQS \(p. 157\)](#)
- [Verify a scaling activity for an Auto Scaling group \(p. 161\)](#)
- [Disable a scaling policy for an Auto Scaling group \(p. 163\)](#)
- [Delete a scaling policy \(p. 164\)](#)
- [Example scaling policies for the AWS Command Line Interface \(AWS CLI\) \(p. 166\)](#)

How dynamic scaling policies work

A dynamic scaling policy instructs Amazon EC2 Auto Scaling to track a specific CloudWatch metric, and it defines what action to take when the associated CloudWatch alarm is in ALARM. The metrics that are

used to invoke the alarm state are an aggregation of metrics coming from all of the instances in the Auto Scaling group. (For example, let's say you have an Auto Scaling group with two instances where one instance is at 60 percent CPU and the other is at 40 percent CPU. On average, they are at 50 percent CPU.) When the policy is in effect, Amazon EC2 Auto Scaling adjusts the group's desired capacity up or down when the threshold of an alarm is breached.

When a dynamic scaling policy is invoked, if the capacity calculation produces a number outside of the minimum and maximum size range of the group, Amazon EC2 Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits. Capacity is measured in one of two ways: using the same units that you chose when you set the desired capacity in terms of instances, or using capacity units (if [instance weighting \(p. 68\)](#) is applied).

- Example 1: An Auto Scaling group has a maximum capacity of 3, a current capacity of 2, and a dynamic scaling policy that adds 3 instances. When invoking this policy, Amazon EC2 Auto Scaling adds only 1 instance to the group to prevent the group from exceeding its maximum size.
- Example 2: An Auto Scaling group has a minimum capacity of 2, a current capacity of 3, and a dynamic scaling policy that removes 2 instances. When invoking this policy, Amazon EC2 Auto Scaling removes only 1 instance from the group to prevent the group from becoming less than its minimum size.

When the desired capacity reaches the maximum size limit, scaling out stops. If demand drops and capacity decreases, Amazon EC2 Auto Scaling can scale out again.

The exception is when you use instance weighting. In this case, Amazon EC2 Auto Scaling can scale out above the maximum size limit, but only by up to your maximum instance weight. Its intention is to get as close to the new desired capacity as possible but still adhere to the allocation strategies that are specified for the group. The allocation strategies determine which instance types to launch. The weights determine how many capacity units each instance contributes to the desired capacity of the group based on its instance type.

- Example 3: An Auto Scaling group has a maximum capacity of 12, a current capacity of 10, and a dynamic scaling policy that adds 5 capacity units. Instance types have one of three weights assigned: 1, 4, or 6. When invoking the policy, Amazon EC2 Auto Scaling chooses to launch an instance type with a weight of 6 based on the allocation strategy. The result of this scale-out event is a group with a desired capacity of 12 and a current capacity of 16.

Multiple dynamic scaling policies

In most cases, a target tracking scaling policy is sufficient to configure your Auto Scaling group to scale out and scale in automatically. A target tracking scaling policy allows you to select a desired outcome and have the Auto Scaling group add and remove instances as needed to achieve that outcome.

For an advanced scaling configuration, your Auto Scaling group can have more than one scaling policy. For example, you can define one or more target tracking scaling policies, one or more step scaling policies, or both. This provides greater flexibility to cover multiple scenarios.

To illustrate how multiple dynamic scaling policies work together, consider an application that uses an Auto Scaling group and an Amazon SQS queue to send requests to a single EC2 instance. To help ensure that the application performs at optimum levels, there are two policies that control when the Auto Scaling group should scale out. One is a target tracking policy that uses a custom metric to add and remove capacity based on the number of SQS messages in the queue. The other is a step scaling policy that uses the Amazon CloudWatch CPUUtilization metric to add capacity when the instance exceeds 90 percent utilization for a specified length of time.

When there are multiple policies in force at the same time, there's a chance that each policy could instruct the Auto Scaling group to scale out (or in) at the same time. For example, it's possible that the CPUUtilization metric spikes and breaches the threshold of the CloudWatch alarm at the same time that the SQS custom metric spikes and breaches the threshold of the custom metric alarm.

When these situations occur, Amazon EC2 Auto Scaling chooses the policy that provides the largest capacity for both scale out and scale in. Suppose, for example, that the policy for CPUUtilization launches one instance, while the policy for the SQS queue launches two instances. If the scale-out criteria for both policies are met at the same time, Amazon EC2 Auto Scaling gives precedence to the SQS queue policy. This results in the Auto Scaling group launching two instances.

The approach of giving precedence to the policy that provides the largest capacity applies even when the policies use different criteria for scaling in. For example, if one policy terminates three instances, another policy decreases the number of instances by 25 percent, and the group has eight instances at the time of scale in, Amazon EC2 Auto Scaling gives precedence to the policy that provides the largest number of instances for the group. This results in the Auto Scaling group terminating two instances (25 percent of 8 = 2). The intention is to prevent Amazon EC2 Auto Scaling from removing too many instances.

We recommend caution, however, when using target tracking scaling policies with step scaling policies because conflicts between these policies can cause undesirable behavior. For example, if the step scaling policy initiates a scale-in activity before the target tracking policy is ready to scale in, the scale-in activity will not be blocked. After the scale-in activity completes, the target tracking policy could instruct the group to scale out again.

Target tracking scaling policies for Amazon EC2 Auto Scaling

To create a target tracking scaling policy, you specify an Amazon CloudWatch metric and a target value that represents the ideal average utilization or throughput level for your application. Amazon EC2 Auto Scaling can then scale out your group (add more instances) to handle peak traffic, and scale in your group (run fewer instances) to reduce costs during periods of low utilization or throughput.

For example, let's say that you currently have an application that runs on two instances, and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This gives you extra capacity to handle traffic spikes without maintaining an excessive number of idle resources.

You can meet this need by creating a target tracking scaling policy that targets an average CPU utilization of 50 percent. Then, your Auto Scaling group scales the number of instances to keep the actual metric value at or near 50 percent.

Topics

- [Multiple target tracking scaling policies \(p. 135\)](#)
- [Considerations \(p. 136\)](#)
- [Choose metrics \(p. 136\)](#)
- [Define target value \(p. 137\)](#)
- [Define instance warm-up time \(p. 138\)](#)
- [Create a target tracking scaling policy \(console\) \(p. 138\)](#)
- [Create a target tracking scaling policy \(AWS CLI\) \(p. 139\)](#)

Multiple target tracking scaling policies

To help optimize scaling performance, you can use multiple target tracking scaling policies together, provided that each of them uses a different metric. For example, utilization and throughput can influence each other. Whenever one of these metrics changes, it usually implies that other metrics will also be impacted. The use of multiple metrics therefore provides additional information about the load that your Auto Scaling group is under and improves decision making when determining how much capacity to add to your group.

The intention of Amazon EC2 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 Auto Scaling group 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.

Considerations

The following considerations apply when working with target tracking scaling policies:

- Do not create, edit, or delete the CloudWatch alarms that are used with a target tracking scaling policy. Amazon EC2 Auto Scaling creates and manages the CloudWatch alarms that are associated with your target tracking scaling policies and deletes them when no longer needed.
- A target tracking scaling policy prioritizes availability during periods of fluctuating traffic levels by scaling in more gradually when traffic is decreasing. If you want your Auto Scaling group to scale in immediately when a workload finishes, you can disable the scale-in portion of the policy. This provides you with the flexibility to use the scale-in method that best meets your needs when utilization is low. To ensure that scale in happens as quickly as possible, we recommend not using a simple scaling policy to prevent a cooldown period from being added.
- If the metric is missing data points, this causes the CloudWatch alarm state to change to INSUFFICIENT_DATA. When this happens, Amazon EC2 Auto Scaling cannot scale your group until new data points are found.
- You might see gaps between the target value and the actual metric data points. This is because we act conservatively by rounding up or down when determining how many instances to add or remove. This prevents us from adding an insufficient number of instances or removing too many instances. However, for smaller Auto Scaling groups with fewer instances, the utilization of the group might seem far from the target value. For example, let's say that you set a target value of 50 percent for CPU utilization and your Auto Scaling group then exceeds the target. We might determine that adding 1.5 instances will decrease the CPU utilization to close to 50 percent. Because it is not possible to add 1.5 instances, we round up and add two instances. This might decrease the CPU utilization to a value below 50 percent, but it ensures that your application has enough resources to support it. Similarly, if we determine that removing 1.5 instances increases your CPU utilization to above 50 percent, we remove just one instance.
- For larger Auto Scaling groups with more instances, the utilization is spread over a larger number of instances, in which case adding or removing instances causes less of a gap between the target value and the actual metric data points.
- A target tracking scaling policy assumes that it should scale out your Auto Scaling group when the specified metric is above the target value. You can't use a target tracking scaling policy to scale out your Auto Scaling group when the specified metric is below the target value.

Choose metrics

In a target tracking scaling policy, you can use predefined or custom metrics.

The following predefined metrics are available:

- ASGAverageCPUUtilization—Average CPU utilization of the Auto Scaling group.
- ASGAverageNetworkIn—Average number of bytes received by a single instance on all network interfaces.
- ASGAverageNetworkOut—Average number of bytes sent out from a single instance on all network interfaces.
- ALBRequestCountPerTarget—Average Application Load Balancer request count per target.

Important

Other valuable information about the metrics for CPU utilization, network I/O, and Application Load Balancer request count per target can be found in the [List the available CloudWatch metrics for your instances](#) topic in the *Amazon EC2 User Guide for Linux Instances* and the [CloudWatch metrics for your Application Load Balancer](#) topic in the *User Guide for Application Load Balancers*, respectively.

You can choose other available CloudWatch metrics or your own metrics in CloudWatch by specifying a custom metric. You must use the AWS CLI or an SDK to create a target tracking policy with a custom metric.

Keep the following in mind when choosing a metric:

- Not all metrics work for target tracking. This can be important when you are specifying a custom metric. The metric must be a valid utilization metric and describe how busy an instance is. The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group. That's so the metric data can be used to proportionally scale out or in the number of instances. For example, the CPU utilization of an Auto Scaling group works (that is, the Amazon EC2 metric `CPUUtilization` with the metric dimension `AutoScalingGroupName`), if the load on the Auto Scaling group is distributed across the instances.
- The following metrics do not work for target tracking:
 - The number of requests received by the load balancer fronting the Auto Scaling group (that is, the Elastic Load Balancing metric `RequestCount`). The number of requests received by the load balancer doesn't change based on the utilization of the Auto Scaling group.
 - Load balancer request latency (that is, the Elastic Load Balancing metric `Latency`). Request latency can increase based on increasing utilization, but doesn't necessarily change proportionally.
 - The CloudWatch Amazon SQS queue metric `ApproximateNumberOfMessagesVisible`. The number of messages in a queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. However, a custom metric that measures the number of messages in the queue per EC2 instance in the Auto Scaling group can work. For more information, see [Scaling based on Amazon SQS \(p. 157\)](#).
- To use the `ALBRequestCountPerTarget` metric, you must specify the `ResourceLabel` parameter to identify the load balancer target group that is associated with the metric.
- When a metric emits real 0 values to CloudWatch (for example, `ALBRequestCountPerTarget`), an Auto Scaling group can scale in to 0 when there is no traffic to your application. To have your Auto Scaling group scale in to 0 when no requests are routed to it, the group's minimum capacity must be set to 0.

When you use EC2 instance metrics in your scaling policies, we recommend that you configure these metrics with a 1-minute granularity to ensure a faster response to changes in the metric value. Scaling on instance metrics with a 5-minute granularity can result in slower response times and scaling on stale metric data.

To get this level of data for Amazon EC2 metrics, you must specifically enable detailed monitoring. By default, EC2 instances are enabled for basic monitoring, which means metric data for instances is available at 5-minute granularity. For more information, see [Configure monitoring for Auto Scaling instances \(p. 265\)](#).

Define target value

When you create a target tracking scaling policy, you must specify a target value. The target value represents the optimal average utilization or throughput for the Auto Scaling group. To use resources cost efficiently, set the target value as high as possible with a reasonable buffer for unexpected traffic increases. When your application is optimally scaled out for a normal traffic flow, the actual metric value should be at or just below the target value.

When a scaling policy is based on throughput, such as the request count per target for an Application Load Balancer, network I/O, or other count metrics, the target value represents the optimal average throughput from a single instance, for a one-minute period.

Define instance warm-up time

Important

We recommend using the default instance warm-up setting, which unifies all the warm-up and cooldown settings for your Auto Scaling group. For more information, see [Available warm-up and cooldown settings \(p. 155\)](#).

You can optionally specify the number of seconds that it takes for a newly launched instance to warm up. Until its specified warm-up time has expired, an instance is not counted toward the aggregated EC2 instance metrics of the Auto Scaling group.

While instances are in the warm-up period, your scaling policies only scale out if the metric value from instances that are not warming up is greater than the policy's target utilization.

If the group scales out again, the instances that are still warming up are counted as part of the desired capacity for the next scale-out activity. The intention is to continuously (but not excessively) scale out.

While the scale-out activity is in progress, all scale-in activities initiated by scaling policies are blocked until the instances finish warming up.

Create a target tracking scaling policy (console)

You can choose to configure a target tracking scaling policy on an Auto Scaling group as you create it or after the Auto Scaling group is created.

To create an Auto Scaling group with a target tracking scaling policy

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Choose **Create Auto Scaling group**.
3. In Steps 1, 2, and 3, choose the options as desired and proceed to **Step 4: Configure group size and scaling policies**.
4. Under **Group size**, specify the range that you want to scale between by updating the minimum capacity and maximum capacity. These two settings allow your Auto Scaling group to scale dynamically. Amazon EC2 Auto Scaling scales your group in the range of values specified by the minimum capacity and maximum capacity.
5. Under **Scaling policies**, choose **Target tracking scaling policy**.
6. To define a policy, do the following:
 - a. Specify a name for the policy.
 - b. For **Metric type**, choose a metric.

If you chose **Application Load Balancer request count per target**, choose a target group in **Target group**.
 - c. Specify a **Target value** for the metric.
 - d. (Optional) For **Instances need**, update the instance warm-up value as needed.
 - e. (Optional) Select **Disable scale in to create only a scale-out policy**. This allows you to create a separate scale-in policy of a different type if wanted.
7. Proceed to create the Auto Scaling group. Your scaling policy will be created after the Auto Scaling group has been created.

To create a target tracking scaling policy for an existing Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. Verify that the minimum capacity and maximum capacity are appropriately set. For example, if your group is already at its maximum size, specify a new maximum in order to scale out. Amazon EC2 Auto Scaling does not scale your group below the minimum capacity or above the maximum capacity. To update your group, on the **Details** tab, change the current settings for minimum and maximum capacity.
4. On the **Automatic scaling** tab, in **Dynamic scaling policies**, choose **Create dynamic scaling policy**.
5. To define a policy, do the following:
 - a. For **Policy type**, leave the default of **Target tracking scaling**.
 - b. Specify a name for the policy.
 - c. For **Metric type**, choose a metric. You can choose only one metric type. To use more than one metric, create multiple policies.

If you chose **Application Load Balancer request count per target**, choose a target group in **Target group**.

 - d. Specify a **Target value** for the metric.
 - e. (Optional) For **Instances need**, update the instance warm-up value as needed.
 - f. (Optional) Select **Disable scale in to create only a scale-out policy**. This allows you to create a separate scale-in policy of a different type if wanted.
6. Choose **Create**.

Create a target tracking scaling policy (AWS CLI)

Use the AWS CLI as follows to configure target tracking scaling policies for your Auto Scaling group.

Tasks

- [Step 1: Create an Auto Scaling group \(p. 139\)](#)
- [Step 2: Create a target tracking scaling policy \(p. 139\)](#)

Step 1: Create an Auto Scaling group

Use the `create-auto-scaling-group` command to create an Auto Scaling group named `my-asg` using the launch template `my-template`. If you don't have a launch template, see [AWS CLI examples for working with launch templates \(p. 33\)](#).

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
--launch-template LaunchTemplateName=my-template,Version='2' \
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
--max-size 5 --min-size 1
```

Step 2: Create a target tracking scaling policy

After you have created the Auto Scaling group, you can create a target tracking scaling policy that instructs Amazon EC2 Auto Scaling to increase and decrease the number of running EC2 instances in the group dynamically when the load on the application changes.

Example: target tracking configuration file

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

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

For more information, see [PredefinedMetricSpecification](#) in the *Amazon EC2 Auto Scaling API Reference*.

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

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

For more information, see [CustomizedMetricSpecification](#) in the *Amazon EC2 Auto Scaling API Reference*.

Example: cpu40-target-tracking-scaling-policy

Use the [put-scaling-policy](#) command, along with the config.json file that you created previously, to create a scaling policy named cpu40-target-tracking-scaling-policy that keeps the average CPU utilization of the Auto Scaling group at 40 percent.

```
aws autoscaling put-scaling-policy --policy-name cpu40-target-tracking-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
  --target-tracking-configuration file://config.json
```

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

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-c665-4bfd-  
  aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-scaling-  
  policy",  
  "Alarms": [  
    {  
      "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-asg-  
      AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",  
      "AlarmName": "TargetTracking-my-asg-AlarmHigh-  
      fc0e4183-23ac-497e-9992-691c9980c38e"  
    }  
  ]  
}
```

```
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-asg-
AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

Step and simple scaling policies for Amazon EC2 Auto Scaling

With step scaling and simple scaling, you choose scaling metrics and threshold values for the CloudWatch alarms that invoke the scaling process. You also define how your Auto Scaling group should be scaled when a threshold is in breach for a specified number of evaluation periods.

We strongly recommend that you use a target tracking scaling policy to scale on a metric like average CPU utilization or the RequestCountPerTarget metric from the Application Load Balancer. Metrics that decrease when capacity increases and increase when capacity decreases can be used to proportionally scale out or in the number of instances using target tracking. This helps ensure that Amazon EC2 Auto Scaling follows the demand curve for your applications closely. For more information, see [Target tracking scaling policies \(p. 135\)](#). You still have the option to use step scaling as an additional policy for a more advanced configuration. For example, you can configure a more aggressive response when demand reaches a certain level.

Contents

- [Differences between step scaling policies and simple scaling policies \(p. 141\)](#)
- [Step adjustments for step scaling \(p. 142\)](#)
- [Scaling adjustment types \(p. 143\)](#)
- [Instance warm-up \(p. 144\)](#)
- [Create CloudWatch alarms for the metric high and low thresholds \(console\) \(p. 145\)](#)
- [Create step scaling policies \(console\) \(p. 146\)](#)
- [Create scaling policies and CloudWatch alarms \(AWS CLI\) \(p. 147\)](#)
 - [Step 1: Create an Auto Scaling group \(p. 148\)](#)
 - [Step 2: Create scaling policies \(p. 148\)](#)
 - [Step scaling policies \(p. 148\)](#)
 - [Simple scaling policies \(p. 149\)](#)
 - [Step 3: Create CloudWatch alarms for the metric high and low thresholds \(p. 149\)](#)

Differences between step scaling policies and simple scaling policies

Step scaling policies and simple scaling policies are two of the dynamic scaling options available for you to use. Both require you to create CloudWatch alarms for the scaling policies. Both require you to specify the high and low thresholds for the alarms. Both require you to define whether to add or remove instances, and how many, or set the group to an exact size.

The main difference between the policy types is the step adjustments that you get with step scaling policies. When *step adjustments* are applied, and they increase or decrease the current capacity of your Auto Scaling group, the adjustments vary based on the size of the alarm breach.

In most cases, step scaling policies are a better choice than simple scaling policies, even if you have only a single scaling adjustment.

The main issue with simple scaling is that after a scaling activity is started, the policy must wait for the scaling activity or health check replacement to complete and the [cooldown period \(p. 153\)](#) to end before responding to additional alarms. Cooldown periods help to prevent the initiation of additional scaling activities before the effects of previous activities are visible.

In contrast, with step scaling the policy can continue 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 Amazon EC2 Auto Scaling as it receives the alarm messages.

Amazon EC2 Auto Scaling originally supported only simple scaling policies. If you created your scaling policy before target tracking and step policies were introduced, your policy is treated as a simple scaling policy.

Step adjustments for step scaling

When you create a step scaling policy, you specify one or more step adjustments that automatically scale the number of instances dynamically 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 that is associated with your CloudWatch alarm. When the alarm is breached, the appropriate scaling policy is invoked. Amazon EC2 Auto Scaling applies the 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 an Auto Scaling group that has both a current capacity and a desired capacity of 10. You have a CloudWatch alarm with a breach threshold of 50 percent and scale-out and scale-in policies. You have a set of step adjustments with an adjustment type of **PercentChangeInCapacity** (or **Percent of group** in the console) for each policy:

Example: Step adjustments for scale-out policy

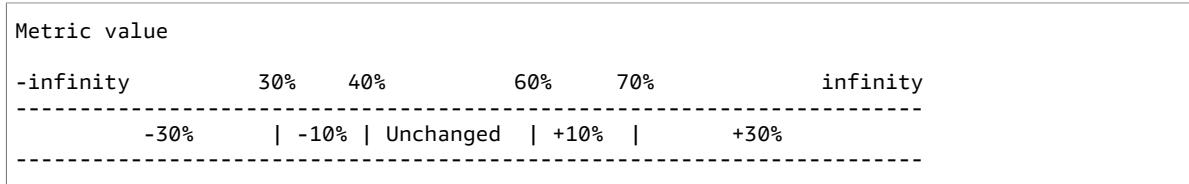
Lower bound	Upper bound	Adjustment
0	10	0
10	20	10
20	null	30

Example: Step adjustments for scale-in policy

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

Lower bound	Upper bound	Adjustment
null	-20	-30

This creates the following scaling configuration.



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

- The desired and current capacity is maintained while the aggregated metric value is greater than 40 and less than 60.
- If the metric value gets to 60, the desired capacity of the group increases by 1 instance, to 11 instances, based on the second step adjustment of the scale-out policy (add 10 percent of 10 instances). After the new instance is running and its specified warm-up time has expired, the current capacity of the group increases to 11 instances. If the metric value rises to 70 even after this increase in capacity, the desired capacity of the group increases by another 3 instances, to 14 instances. This is based on the third step adjustment of the scale-out policy (add 30 percent of 11 instances, 3.3 instances, rounded down to 3 instances).
- If the metric value gets to 40, the desired capacity of the group decreases by 1 instance, to 13 instances, based on the second step adjustment of the scale-in policy (remove 10 percent of 14 instances, 1.4 instances, rounded down to 1 instance). If the metric value falls to 30 even after this decrease in capacity, the desired capacity of the group decreases by another 3 instances, to 10 instances. This is based on the third step adjustment of the scale-in policy (remove 30 percent of 13 instances, 3.9 instances, rounded down to 3 instances).

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

- If you are using the AWS Management Console, you specify the upper and lower bounds as absolute values. If you are using the AWS CLI or an SDK, you specify the upper and lower bounds relative to the breach threshold.
- 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 Auto Scaling group, or in capacity units. Normally a capacity unit means one instance, unless you are using the instance weighting feature.

Amazon EC2 Auto Scaling supports the following adjustment types for step scaling and simple scaling:

- **ChangeInCapacity** — Increment or decrement the current capacity of the group by the specified value. A positive value increases the capacity and a negative adjustment value decreases the capacity. For example: If the current capacity of the group is 3 and the adjustment is 5, then when this policy is performed, we add 5 capacity units to the capacity for a total of 8 capacity units.
- **ExactCapacity** — Change the current capacity of the group to the specified value. Specify a positive value with this adjustment type. For example: If the current capacity of the group is 3 and the adjustment is 5, then when this policy is performed, we change the capacity to 5 capacity units.
- **PercentChangeInCapacity** — Increment or decrement the current capacity of the group 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 when this policy is performed, we add 1 capacity unit to the capacity for a total of 11 capacity units.

Note

If the resulting value is not an integer, it is rounded 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 number of instances to scale using the **MinAdjustmentMagnitude** parameter. For example, suppose that you create a policy that adds 25 percent and you specify a minimum increment of 2 instances. If you have an Auto Scaling group with 4 instances and the scaling policy is executed, 25 percent of 4 is 1 instance. However, because you specified a minimum increment of 2, there are 2 instances added.

When [instance weighting \(p. 68\)](#) is used, the effect of setting the **MinAdjustmentMagnitude** parameter to a non-zero value changes. The value is in capacity units. To set the minimum number of instances to scale, set this parameter to a value that is at least as large as your largest instance weight.

If you are using instance weighting, keep in mind that the current capacity of your Auto Scaling group can exceed the desired capacity as needed. If your absolute number to decrement, or the amount that the percentage says to decrement, is less than the difference between current and desired capacity, no scaling action is taken. You must take these behaviors into account when you look at the outcome of a scaling policy when a threshold alarm is in breach. For example, suppose that the desired capacity is 30 and the current capacity is 32. When the alarm is in breach, if the scaling policy decrements the desired capacity by 1, then no scaling action is taken.

Instance warm-up

Important

We recommend using the **DefaultInstanceWarmup** setting, which unifies all the warm-up and cooldown settings for your Auto Scaling group. For more information, see [Available warm-up and cooldown settings \(p. 155\)](#).

For step scaling, you can optionally specify the number of seconds that it takes for a newly launched instance to warm up. Until its specified warm-up time has expired, an instance is not counted toward the aggregated EC2 instance metrics of the Auto Scaling group.

While instances are in the warm-up period, your scaling policies only scale out if the metric value from instances that are not warming up is greater than the policy's alarm high threshold.

If the group scales out again, the instances that are still warming up are counted as part of the desired capacity for the next scale-out activity. Therefore, multiple alarm breaches that fall in the range of the same step adjustment result in a single scaling activity. The intention is to continuously (but not excessively) scale out.

While the scale-out activity is in progress, all scale-in activities initiated by scaling policies are blocked until the instances finish warming up.

Create CloudWatch alarms for the metric high and low thresholds (console)

You can use the CloudWatch console to create two alarms, one for scaling out (metric high) and the other for scaling in (metric low). When the threshold of an alarm is breached, for example, because the traffic has increased, the alarm goes into the ALARM state. This state change invokes the scaling policy associated with the alarm. The policy then instructs Amazon EC2 Auto Scaling how to respond to the alarm breach, such as by adding or removing a specified number of instances.

Example: To create a CloudWatch alarm for the metric high threshold

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region. From the navigation bar, select the Region where your Auto Scaling group resides.
3. In the navigation pane, choose **Alarms, All alarms** and then choose **Create alarm**.
4. Choose **Select metric**.
5. On the **All metrics** tab, choose **EC2, By Auto Scaling Group**, and enter the Auto Scaling group's name in the search field. Then, select **CPUUtilization** and choose **Select metric**. The **Specify metric and conditions** page appears, showing a graph and other information about the metric.
6. For **Period**, choose the evaluation period for the alarm, for example, 1 minute. When evaluating the alarm, each period is aggregated into one data point.

Note

A shorter period creates a more sensitive alarm.

7. Under **Conditions**, do the following:
 - For **Threshold type**, choose **Static**.
 - For **Whenever CPUUtilization is**, specify whether you want the value of the metric to be greater than, greater than or equal to, less than, or less than or equal to the threshold to breach the alarm. Then, under **than**, enter the threshold value that you want to breach the alarm.

Important

For an alarm to use with a scale-out policy (metric high), make sure you do not choose less than or less than or equal to the threshold. For an alarm to use with a scale-in policy (metric low), make sure you do not choose greater than or greater than or equal to the threshold.

8. Under **Additional configuration**, do the following:
 - For **Datapoints to alarm**, enter the number of data points (evaluation periods) during which the metric value must meet the threshold conditions for the alarm. For example, two consecutive periods of 5 minutes would take 10 minutes to invoke the alarm state.
 - For **Missing data treatment**, choose **Treat missing data as bad (breaching threshold)**. For more information, see [Configuring how CloudWatch alarms treat missing data](#) in the *Amazon CloudWatch User Guide*.
9. Choose **Next**.

The **Configure actions** page appears.

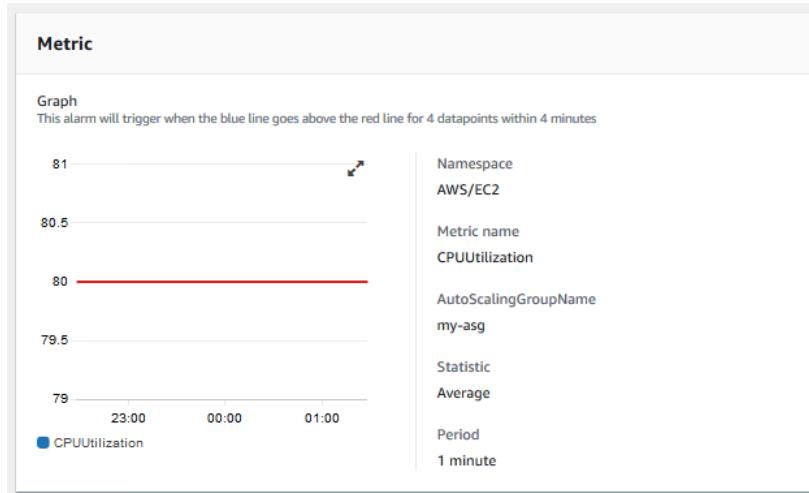
10. Under **Notification**, select an Amazon SNS topic to notify when the alarm is in ALARM state, OK state, or INSUFFICIENT_DATA state.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.

11. You can leave the other sections of the **Configure actions** page empty. Leaving the other sections empty creates an alarm without associating it to a scaling policy. You can then associate the alarm with a scaling policy from the Amazon EC2 Auto Scaling console.
12. Choose **Next**.
13. Enter a name (for example, Step-Scaling-AlarmHigh-AddCapacity) and, optionally, a description for the alarm, and then choose **Next**.
14. Choose **Create alarm**.

Example: CloudWatch alarm that breaches if CPU goes above the 80 percent threshold for 4 minutes



Create step scaling policies (console)

You can choose to configure a step scaling policy on an Auto Scaling group after the group is created.

The following procedure shows you how to use the Amazon EC2 Auto Scaling console to create two step scaling policies: a scale-out policy that increases the capacity of the group by 30 percent, and a scale-in policy that decreases the capacity of the group to two instances.

While you are configuring your scaling policies, you can create the alarms at the same time. Alternatively, you can use alarms that you created from the CloudWatch console, as described in the previous section.

To create a step scaling policy for scale out

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. Verify that the minimum and maximum size limits are appropriately set. For example, if your group is already at its maximum size, you need to specify a new maximum in order to scale out. Amazon EC2 Auto Scaling does not scale your group below the minimum capacity or above the maximum capacity. To update your group, on the **Details** tab, change the current settings for minimum and maximum capacity.
4. On the **Automatic scaling** tab, in **Dynamic scaling policies**, choose **Create dynamic scaling policy**.
5. To define a policy for scale out (increase capacity), do the following:

- a. For **Policy type**, choose **Step scaling**.
- b. Specify a name for the policy.
- c. For **CloudWatch alarm**, choose your alarm. If you haven't already created an alarm, choose [Create a CloudWatch alarm](#) and complete step 4 through step 14 in [Create CloudWatch alarms for the metric high and low thresholds \(console\) \(p. 145\)](#) to create an alarm that monitors CPU utilization. Set the alarm threshold to greater than or equal to 80 percent.
- d. Specify the change in the current group size that this policy will make when executed using **Take the action**. You can add a specific number of instances or a percentage of the existing group size, or set the group to an exact size.

For example, choose Add, enter 30 in the next field, and then choose **percent of group**. By default, the lower bound for this step adjustment is the alarm threshold and the upper bound is positive (+) infinity.

- e. To add another step, choose **Add step** and then define the amount by which to scale and the lower and upper bounds of the step relative to the alarm threshold.
- f. To set a minimum number of instances to scale, update the number field in **Add capacity units in increments of at least 1 capacity units**.
- g. (Optional) For **Instances need**, update the instance warm-up value as needed.

6. Choose **Create**.

To create a step scaling policy for scale in

1. Choose **Create dynamic scaling policy** to continue where you left off after creating a policy for scale out.
2. To define a policy for scale in (decrease capacity), do the following:
 - a. For **Policy type**, choose **Step scaling**.
 - b. Specify a name for the policy.
 - c. For **CloudWatch alarm**, choose your alarm. If you haven't already created an alarm, choose [Create a CloudWatch alarm](#) and complete step 4 through step 14 in [Create CloudWatch alarms for the metric high and low thresholds \(console\) \(p. 145\)](#) to create an alarm that monitors CPU utilization. Set the alarm threshold to less than or equal to 40 percent.
 - d. Specify the change in the current group size that this policy will make when executed using **Take the action**. You can remove a specific number of instances or a percentage of the existing group size, or set the group to an exact size.

For example, choose Remove, enter 2 in the next field, and then choose **capacity units**. By default, the upper bound for this step adjustment is the alarm threshold and the lower bound is negative (-) infinity.

3. Choose **Create**.
- e. To add another step, choose **Add step** and then define the amount by which to scale and the lower and upper bounds of the step relative to the alarm threshold.

Create scaling policies and CloudWatch alarms (AWS CLI)

Use the AWS CLI as follows to configure step or simple scaling policies for your Auto Scaling group.

Tasks

- [Step 1: Create an Auto Scaling group \(p. 148\)](#)
- [Step 2: Create scaling policies \(p. 148\)](#)
- [Step 3: Create CloudWatch alarms for the metric high and low thresholds \(p. 149\)](#)

Step 1: Create an Auto Scaling group

Use the following [create-auto-scaling-group](#) command to create an Auto Scaling group named `my-asg` using the launch template `my-template`. If you don't have a launch template, see [AWS CLI examples for working with launch templates \(p. 33\)](#).

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
  --launch-template LaunchTemplateName=my-template,Version='2' \  
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
  --max-size 5 --min-size 1
```

Step 2: Create scaling policies

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

Step scaling policies

Example: my-step-scale-out-policy

Use the following [put-scaling-policy](#) command to create a step scaling policy named `my-step-scale-out-policy`, with an adjustment type of `PercentChangeInCapacity` that increases the capacity of the group based on the following step adjustments (assuming a CloudWatch alarm threshold of 60 percent):

- Increase the instance count by 10 percent when the value of the metric is greater than or equal to 60 percent but less than 75 percent
- Increase the instance count by 20 percent when the value of the metric is greater than or equal to 75 percent but less than 85 percent
- Increase the instance count by 30 percent when the value of the metric is greater than or equal to 85 percent

```
aws autoscaling put-scaling-policy \  
  --auto-scaling-group-name my-asg \  
  --policy-name my-step-scale-out-policy \  
  --policy-type StepScaling \  
  --adjustment-type PercentChangeInCapacity \  
  --metric-aggregation-type Average \  
  --step-adjustments \  
    MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \  
    MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \  
      MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \  
  --min-adjustment-magnitude 1
```

Record the policy's Amazon Resource Name (ARN). You need it to create a CloudWatch alarm for the policy.

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-  
  b53b-aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy  
}
```

Example: my-step-scale-in-policy

Use the following [put-scaling-policy](#) command to create a step scaling policy named `my-step-scale-in-policy`, with an adjustment type of `ChangeInCapacity` that decreases the capacity of the group by 2 instances when the associated CloudWatch alarm breaches the metric low threshold value.

```
aws autoscaling put-scaling-policy \
--auto-scaling-group-name my-asg \
--policy-name my-step-scale-in-policy \
--policy-type StepScaling \
--adjustment-type ChangeInCapacity \
--step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

Record the policy's Amazon Resource Name (ARN). You need it to create the CloudWatch alarm for the policy.

```
{  
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy"  
}
```

Simple scaling policies

Alternatively, you can create simple scaling policies by using the following CLI commands instead of the preceding CLI commands. Keep in mind that a cooldown period will be in place due to the use of simple scaling policies.

Example: *my-simple-scale-out-policy*

Use the following [put-scaling-policy](#) command to create a simple scaling policy named *my-simple-scale-out-policy*, with an adjustment type of *PercentChangeInCapacity* that increases the capacity of the group by 30 percent when the associated CloudWatch alarm breaches the metric high threshold value.

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
--auto-scaling-group-name my-asg --scaling-adjustment 30 \
--adjustment-type PercentChangeInCapacity
```

Record the policy's Amazon Resource Name (ARN). You need it to create the CloudWatch alarm for the policy.

Example: *my-simple-scale-in-policy*

Use the following [put-scaling-policy](#) command to create a simple scaling policy named *my-simple-scale-in-policy*, with an adjustment type of *ChangeInCapacity* that decreases the capacity of the group by one instance when the associated CloudWatch alarm breaches the metric low threshold value.

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
--auto-scaling-group-name my-asg --scaling-adjustment -1 \
--adjustment-type ChangeInCapacity --cooldown 180
```

Record the policy's Amazon Resource Name (ARN). You need it to create the CloudWatch alarm for the policy.

Step 3: Create CloudWatch alarms for the metric high and low thresholds

In step 2, you created scaling policies that provided instructions to Amazon EC2 Auto Scaling about how to scale out and scale in when a metric's value is increasing or decreasing. In this step, you create two alarms by identifying the metric to watch, defining the metric high and low thresholds and other details for the alarms, and associating the alarms with the scaling policies.

Example: *AddCapacity*

Use the following CloudWatch [put-metric-alarm](#) command to create an alarm that increases the size of the Auto Scaling group based on an average CPU threshold value of 60 percent for at least two

consecutive evaluation periods of two minutes. To 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-AddCapacity \  
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \  
  --period 120 --evaluation-periods 2 --threshold 60 \  
  --comparison-operator GreaterThanOrEqualToThreshold \  
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \  
  --alarm-actions PolicyARN
```

Example: RemoveCapacity

Use the following CloudWatch `put-metric-alarm` command to create an alarm that decreases the size of the Auto Scaling group based on average CPU threshold value of 40 percent for at least two consecutive evaluation periods of two minutes. To 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-AlarmLow-RemoveCapacity \  
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \  
  --period 120 --evaluation-periods 2 --threshold 40 \  
  --comparison-operator LessThanOrEqualToThreshold \  
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \  
  --alarm-actions PolicyARN
```

Set default values for instance warmup or scaling cooldown

The default instance warmup and the default cooldown are Auto Scaling group properties that you can use to fine-tune scaling performance.

Contents

- [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#)
- [Scaling cooldowns for Amazon EC2 Auto Scaling \(p. 153\)](#)
- [Available warm-up and cooldown settings \(p. 155\)](#)

Set the default instance warmup for an Auto Scaling group

You can use the default instance warmup feature to improve the Amazon CloudWatch metrics used for dynamic scaling. CloudWatch collects and aggregates usage data, such as CPU and network I/O, across your Auto Scaling instances. You use these metrics to create scaling policies that adjust the number of instances in your Auto Scaling group as the selected metric's value increases and decreases.

When default instance warmup is not enabled, each instance starts contributing usage data to the aggregated metrics as soon as the instance reaches the `InService` state. However, if you enable default instance warmup, this lets your instances finish warming up before they contribute the usage data. This keeps dynamic scaling from being affected by metrics for individual instances that aren't handling application traffic yet and that might be experiencing temporarily high usage of compute resources.

Default instance warmup is not configured and is not enabled by default. To optimize the performance of scaling policies that scale continuously, such as target tracking and step scaling policies, we strongly recommend that you enable the default instance warmup, *even if its value is set to 0 seconds*. When default instance warmup is not enabled, the default cooldown might be applied when instances launch, and dynamic scaling scale-in activities will be blocked until the default cooldown period ends.

In addition, by enabling the default instance warmup feature, you no longer have to specify values for warm-up parameters for individual scaling policies and instance refreshes. You can use the default

instance warmup to unify these settings at the group level. By default, these parameters will be set to the value of the default instance warmup. For more information, see [Available warm-up and cooldown settings \(p. 155\)](#).

Contents

- [Choose a time value for the default instance warmup \(p. 151\)](#)
- [Understand how the default instance warmup affects scaling \(p. 151\)](#)
- [Enable the default instance warmup for a group \(p. 151\)](#)
- [Verify the default instance warmup for a group \(p. 152\)](#)

Choose a time value for the default instance warmup

When you choose a warm-up time value, aim for an optimal balance between collecting usage data for legitimate traffic, but minimizing data collection associated with temporary usage spikes on startup.

When instances finish launching, they're attached to the Auto Scaling group and registered to an Elastic Load Balancing load balancer, if used, before they enter the `InService` state. After the instances enter the `InService` state, resource consumption can still experience temporary spikes and need time to stabilize. For example, resource consumption for an application server that must download and cache large assets takes longer to stabilize than a lightweight web server with no large assets to download. The default instance warmup provides the time delay necessary for resource consumption to stabilize.

Important

You should also consider using lifecycle hooks for use cases where you have configuration tasks or scripts to run on startup. Lifecycle hooks can delay instances from being put in service during a scale-out event until they have finished initializing. They are particularly useful if you have bootstrapping scripts that take a while to complete. Therefore, if you add a lifecycle hook, you can reduce the value of the default instance warmup. If you set the default instance warmup to 0 seconds, no additional waiting period is imposed beyond the time needed for the lifecycle hook. For more information about using lifecycle hooks, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Understand how the default instance warmup affects scaling

After you enable the default instance warmup, each newly launched instance in your Auto Scaling group has a warm-up period.

While instances are in the warm-up period, your scaling policies only scale out if the metric value from instances that are not warming up is greater than the policy's alarm high threshold (which is the same as the target utilization of a target tracking scaling policy).

If instances are still warming up and the group scales out again, the instances are counted as part of the desired capacity for the next scale-out activity. Therefore, multiple alarm breaches that fall in the range of the same step adjustment result in a single scaling activity. The intention is to continuously (but not excessively) scale out.

If demand decreases, the intention is to scale in conservatively to protect your application's availability. This blocks scale-in activities initiated by scaling policies until the instances finish warming up.

Enable the default instance warmup for a group

You can enable the default instance warmup when you create an Auto Scaling group. You can also enable it for existing groups.

Console

To enable the default instance warmup for a new group (console)

When you create the Auto Scaling group, on the **Configure advanced options** page, under **Additional settings**, select the **Enable default instance warmup** option. Choose the warm-up time that you need for your application.

AWS CLI

To enable the default instance warmup for a new group (AWS CLI)

To enable the default instance warmup for an Auto Scaling group, add the `--default-instance-warmup` option and specify a value, in seconds, from 0 to 3600. After it's enabled, a value of -1 will turn this setting off.

The following `create-auto-scaling-group` command creates an Auto Scaling group with the name `my-asg` and enables the default instance warmup with a value of `120` seconds.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --default-instance-warmup 120 ...
```

Tip

If this command throws an error, make sure that you have updated the AWS CLI locally to the latest version.

Console

To enable the default instance warmup for an existing group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. On the navigation bar at the top of the screen, choose the AWS Region that you created your Auto Scaling group in.
3. Select the check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
4. On the **Details** tab, choose **Advanced configurations**, **Edit**.
5. For **Default instance warmup**, choose the warm-up time that you need for your application.
6. Choose **Update**.

AWS CLI

To enable the default instance warmup for an existing group (AWS CLI)

The following example uses the `update-auto-scaling-group` command to enable the default instance warmup with a value of `120` seconds for an existing Auto Scaling group named `my-asg`.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --default-instance-warmup 120
```

Tip

If this command throws an error, make sure that you have updated the AWS CLI locally to the latest version.

Verify the default instance warmup for a group

To verify the default instance warmup for an Auto Scaling group (AWS CLI)

Use the `describe-auto-scaling-groups` command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
    "AutoScalingGroups": [  
        {  
            "AutoScalingGroupName": "my-asg",  
            "AutoScalingGroupARN": "arn",  
            ...  
            "DefaultInstanceWarmup": 120  
        }  
    ]  
}
```

Scaling cooldowns for Amazon EC2 Auto Scaling

After your Auto Scaling group launches or terminates instances, it waits for a cooldown period to end before any further scaling activities initiated by simple scaling policies can start. The intention of the cooldown period is to prevent your Auto Scaling group from launching or terminating additional instances before the effects of previous activities are visible.

Important

As a best practice, we recommend that you do not use simple scaling policies and scaling cooldowns.

In most cases, a target tracking scaling policy or a step scaling policy is better for scaling performance. For a scaling policy that changes the size of your Auto Scaling group proportionally as the value of the scaling metric decreases or increases, we recommend [target tracking \(p. 135\)](#) over either simple scaling or step scaling.

Suppose, for example, that a simple scaling policy for CPU utilization recommends launching two instances. Amazon EC2 Auto Scaling launches two instances and then pauses the scaling activities until the cooldown period ends. After the cooldown period ends, any scaling activities initiated by simple scaling policies can resume. If CPU utilization breaches the alarm high threshold again, the Auto Scaling group scales out again, and the cooldown period takes effect again. However, if two instances were enough to bring the metric value back down, the group remains at its current size.

Contents

- [Considerations \(p. 153\)](#)
- [Lifecycle hooks cause additional delays \(p. 154\)](#)
- [Change the default cooldown period \(p. 154\)](#)
- [Set a cooldown period for specific simple scaling policies \(p. 154\)](#)

Considerations

The following considerations apply when working with simple scaling policies and scaling cooldowns:

- Target tracking and step scaling policies can initiate a scale-out activity immediately without waiting for the cooldown period to end. Instead, whenever your Auto Scaling group launches instances, the individual instances have a warm-up period. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).
- When a scheduled action starts at the scheduled time, it can also initiate a scaling activity immediately without waiting for the cooldown period to end.
- If an instance becomes unhealthy, Amazon EC2 Auto Scaling does not wait for the cooldown period to end before replacing the unhealthy instance.

- When multiple instances launch or terminate, the cooldown period (either the default cooldown or the scaling policy-specific cooldown) takes effect starting when the last instance finishes launching or terminating.
- When you manually scale your Auto Scaling group, the default is not to wait for a cooldown to end. However, you can override this behavior and honor the default cooldown when you use the AWS CLI or an SDK to manually scale.
- By default, Elastic Load Balancing waits 300 seconds to complete the deregistration (connection draining) process. If the group is behind an Elastic Load Balancing load balancer, it will wait for the terminating instances to deregister before starting the cooldown period.

Lifecycle hooks cause additional delays

If a [lifecycle hook \(p. 195\)](#) is invoked, the cooldown period begins after you complete the lifecycle action or after the timeout period ends. For example, consider an Auto Scaling group that has a lifecycle hook for instance launch. When the application experiences an increase in demand, the group launches an instance to add capacity. Because there is a lifecycle hook, the instance is put into a wait state and scaling activities due to simple scaling policies are paused. When the instance enters the `InService` state, the cooldown period starts. When the cooldown period ends, simple scaling policy activities are resumed.

When Elastic Load Balancing is enabled, for the purposes of scaling in, the cooldown period starts when the terminating instance finishes connection draining (deregistration delay). It doesn't wait for the hook duration. This means that any scaling activities due to simple scaling policies can resume as soon as the result of the scale-in event is reflected in the capacity of the group. Otherwise, waiting to complete all three activities—connection draining, a lifecycle hook, and a cooldown period—would significantly increase the amount of time that the Auto Scaling group needs to pause scaling.

Change the default cooldown period

You can't set the default cooldown when you initially create an Auto Scaling group in the Amazon EC2 Auto Scaling console. By default, this cooldown period is set to 300 seconds (5 minutes). If needed, you can update this after the group is created.

To change the default cooldown period (console)

After creating the Auto Scaling group, on the **Details** tab, choose **Advanced configurations, Edit**. For **Default cooldown**, choose the amount of time that you want based on your instance startup time or other application needs.

To change the default cooldown period (AWS CLI)

Use the following commands to change the default cooldown for new or existing Auto Scaling groups. If the default cooldown is not defined, the default value of 300 seconds is used.

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

To confirm the value of the default cooldown, use the [describe-auto-scaling-groups](#) command.

Set a cooldown period for specific simple scaling policies

By default, all simple scaling policies use the default cooldown period that is defined for the Auto Scaling group. To set a cooldown period for specific simple scaling policies, use the optional `cooldown` parameter when you create or update the policy. When a cooldown period is specified for a policy, it overrides the default cooldown.

One common use for a scaling policy-specific cooldown period is with a scale-in policy. Because this policy terminates instances, Amazon EC2 Auto Scaling needs less time to determine whether to terminate additional instances. Terminating instances should be a much quicker operation than launching instances. The default cooldown period of 300 seconds is therefore too long. In this case, a scaling policy-specific cooldown period with a lower value for your scale-in policy can help you reduce costs by allowing the group to scale in faster.

To create or update simple scaling policies in the console, choose the **Automatic scaling** tab after you create the group. To create or update simple scaling policies using the AWS CLI, use the [put-scaling-policy](#) command. For more information, see [Step and simple scaling policies \(p. 141\)](#).

Available warm-up and cooldown settings

To help optimize scaling performance, choose the appropriate warm-up and cooldown settings for your Auto Scaling group.

We recommend using the `DefaultInstanceWarmup` setting, which unifies all the warm-up and cooldown settings. If needed, you can also use the `HealthCheckGracePeriod` setting.

The other available settings are not intended to be used at the same time as the `DefaultInstanceWarmup` setting. For example, `EstimatedInstanceWarmup` and `InstanceWarmup` aren't needed when `DefaultInstanceWarmup` is already defined, and `DefaultCooldown` and `Cooldown` are only needed when you use simple scaling policies. (As a best practice, we recommend target tracking and step scaling policies instead of simple scaling policies.) However, we will continue to support all of these settings so that you can choose the appropriate settings for your use case.

[DefaultInstanceWarmup \(Recommended\)](#)

API operation: [CreateAutoScalingGroup](#), [UpdateAutoScalingGroup](#)

The amount of time, in seconds, until a newly launched instance can contribute to the Amazon CloudWatch metrics. This delay lets an instance finish initializing before Amazon EC2 Auto Scaling aggregates instance metrics, resulting in more reliable usage data. Set this value equal to the amount of time that it takes for resource consumption to become stable after an instance reaches the `InService` state. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).

You can reduce the value of the instance warmup if you use a lifecycle hook for instance launch. If your Auto Scaling group is behind a load balancer, you can add a lifecycle hook to the group to make sure that your instances are ready to serve traffic before they are registered to the load balancer at the end of the lifecycle hook. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Important

To manage your warm-up settings at the group level, we recommend that you set the default instance warmup, *even if its value is set to 0 seconds*. This also optimizes the performance of scaling policies that scale continuously, such as target tracking and step scaling policies. If you need to remove a value that you previously set, include the property but specify `-1` for the value. However, we strongly recommend keeping the default instance warmup enabled by specifying a minimum value of `0`.

Default: None

[EstimatedInstanceWarmup](#)

Not needed if DefaultInstanceWarmup is defined.

API operation: [PutScalingPolicy](#)

The estimated time, in seconds, until a newly launched instance can contribute to the CloudWatch metrics. This warm-up period applies to instances launched due to a specific target tracking or step scaling policy. When a warm-up period is specified here, it overrides the default instance warmup.

For more information, see [Target tracking scaling policies \(p. 135\)](#) and [Step and simple scaling policies \(p. 141\)](#).

Default: If not defined, the value of this parameter defaults to the value for the default instance warmup defined for the group. If default instance warmup is null, then it falls back to the value of default cooldown.

InstanceWarmup

Not needed if DefaultInstanceWarmup is defined.

API operation: [StartInstanceRefresh](#)

A warm-up period, in seconds, that applies to a specific instance refresh operation. Specifying a warm-up period when you start an instance refresh overrides the default instance warmup, but only for the current instance refresh. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Default: If not defined, the value of this parameter defaults to the value for the default instance warmup defined for the group. If default instance warmup is null, then it falls back to the value of the health check grace period.

DefaultCooldown

Only needed if you use simple scaling policies.

API operation: [CreateAutoScalingGroup](#), [UpdateAutoScalingGroup](#)

The amount of time, in seconds, between one scaling activity ending and another one starting due to simple scaling policies. For more information, see [Scaling cooldowns for Amazon EC2 Auto Scaling \(p. 153\)](#).

Default: 300 seconds

Cooldown

Only needed if you use simple scaling policies.

API operation: [PutScalingPolicy](#)

A cooldown period, in seconds, that applies to a specific simple scaling policy. (Simple scaling policies are no longer recommended. As a best practice, we recommend target tracking and step scaling policies instead.) When a cooldown period is specified here, it overrides the default cooldown. For more information, see [Scaling cooldowns for Amazon EC2 Auto Scaling \(p. 153\)](#).

Default: None

HealthCheckGracePeriod

API operation: [CreateAutoScalingGroup](#), [UpdateAutoScalingGroup](#)

The amount of time, in seconds, that Amazon EC2 Auto Scaling waits before checking the health status of an EC2 instance that has come into service and marking it unhealthy due to a failed Elastic Load Balancing or custom health check. This is useful if your instances do not immediately pass these health checks after they enter the InService state. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).

You can set the value of the health check grace period to 0 if you use a lifecycle hook for instance launch. If your Auto Scaling group is behind a load balancer, consider adding a lifecycle hook to the group to make sure that your instances are ready to serve traffic before they are registered to the load balancer at the end of the lifecycle hook. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Default: 300 seconds for an Auto Scaling group that you create in the AWS Management Console. 0 seconds for an Auto Scaling group that you create using the AWS CLI or an SDK.

Scaling based on Amazon SQS

This section shows you how to scale your Auto Scaling group in response to changes in system load in an Amazon Simple Queue Service (Amazon SQS) queue. To learn more about how you can use Amazon SQS, see the [Amazon Simple Queue Service Developer Guide](#).

There are some scenarios where you might think about scaling in response to activity in an Amazon SQS queue. For example, suppose that you have a web app that lets users upload images and use them online. In this scenario, each image requires resizing and encoding before it can be published. The app runs on EC2 instances in an Auto Scaling group, and it's configured to handle your typical upload rates. Unhealthy instances are terminated and replaced to maintain current instance levels at all times. The app places the raw bitmap data of the images in an SQS queue for processing. It processes the images and then publishes the processed images where they can be viewed by users. The architecture for this scenario works well if the number of image uploads doesn't vary over time. But if the number of uploads changes over time, you might consider using dynamic scaling to scale the capacity of your Auto Scaling group.

Contents

- [Use target tracking with the right metric \(p. 157\)](#)
- [Limitations and prerequisites \(p. 158\)](#)
- [Configure scaling based on Amazon SQS \(p. 159\)](#)
- [Amazon SQS and instance scale-in protection \(p. 161\)](#)

Use target tracking with the right metric

If you use a target tracking scaling policy based on a custom Amazon SQS queue metric, dynamic scaling can adjust to the demand curve of your application more effectively. For more information about choosing metrics for target tracking, see [Choose metrics \(p. 136\)](#).

The issue with using a CloudWatch Amazon SQS metric like `ApproximateNumberOfMessagesVisible` for target tracking is that the number of messages in the queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. That's because the number of messages in your SQS queue does not solely define the number of instances needed. The number of instances in your Auto Scaling group can be driven by multiple factors, including how long it takes to process a message and the acceptable amount of latency (queue delay).

The solution is to use a *backlog per instance* metric with the target value being the *acceptable backlog per instance* to maintain. You can calculate these numbers as follows:

- **Backlog per instance:** To calculate your backlog per instance, start with the `ApproximateNumberOfMessages` queue attribute to determine the length of the SQS queue (number of messages available for retrieval from the queue). Divide that number by the fleet's running capacity, which for an Auto Scaling group is the number of instances in the `InService` state, to get the backlog per instance.
- **Acceptable backlog per instance:** To calculate your target value, first determine what your application can accept in terms of latency. Then, take the acceptable latency value and divide it by the average time that an EC2 instance takes to process a message.

As an example, let's say that you currently have an Auto Scaling group with 10 instances and the number of visible messages in the queue (`ApproximateNumberOfMessages`) is 1500. If the average processing time is 0.1 seconds for each message and the longest acceptable latency is 10 seconds, then

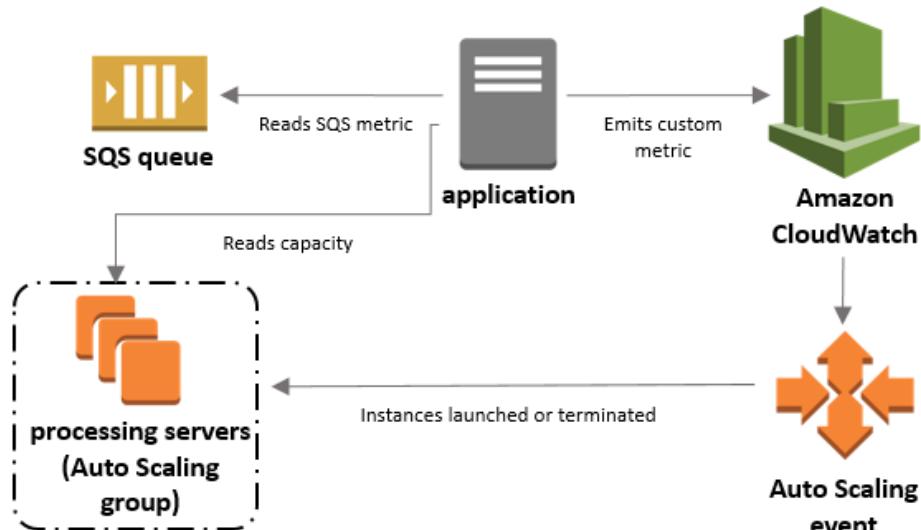
the acceptable backlog per instance is $10 / 0.1$, which equals 100 messages. This means that 100 is the target value for your target tracking policy. When the backlog per instance reaches the target value, a scale-out event will happen. Because the backlog per instance is already 150 messages (1500 messages / 10 instances), your group scales out, and it scales out by five instances to maintain proportion to the target value.

The following procedures demonstrate how to publish the custom metric and create the target tracking scaling policy that configures your Auto Scaling group to scale based on these calculations.

There are three main parts to this configuration:

- An Auto Scaling group to manage EC2 instances for the purposes of processing messages from an SQS queue.
- A custom metric to send to Amazon CloudWatch that measures the number of messages in the queue per EC2 instance in the Auto Scaling group.
- A target tracking policy that configures your Auto Scaling group to scale based on the custom metric and a set target value. CloudWatch alarms invoke the scaling policy.

The following diagram illustrates the architecture of this configuration.



Limitations and prerequisites

To use this configuration, you need to be aware of the following limitations:

- You must use the AWS CLI or an SDK to publish your custom metric to CloudWatch. You can then monitor your metric with the AWS Management Console.
- The Amazon EC2 Auto Scaling console does not support target tracking scaling policies that use custom metrics. You must use the AWS CLI or an SDK to specify a custom metric for your scaling policy.

The following sections direct you to use the AWS CLI for the tasks you need to perform. For example, to get metric data that reflects the present use of the queue, you use the SQS [get-queue-attributes](#) command. Make sure that you have the CLI [installed](#) and [configured](#).

Before you begin, you must have an Amazon SQS queue to use. The following sections assume that you already have a queue (standard or FIFO), an Auto Scaling group, and EC2 instances running the application that uses the queue. For more information about Amazon SQS, see the [Amazon Simple Queue Service Developer Guide](#).

Configure scaling based on Amazon SQS

Tasks

- [Step 1: Create a CloudWatch custom metric \(p. 159\)](#)
- [Step 2: Create a target tracking scaling policy \(p. 159\)](#)
- [Step 3: Test your scaling policy \(p. 160\)](#)

Step 1: Create a CloudWatch custom metric

A custom metric is defined using a metric name and namespace of your choosing. Namespaces for custom metrics cannot start with AWS/. For more information about publishing custom metrics, see the [Publish custom metrics](#) topic in the *Amazon CloudWatch User Guide*.

Follow this procedure to create the custom metric by first reading information from your AWS account. Then, calculate the backlog per instance metric, as recommended in an earlier section. Lastly, publish this number to CloudWatch at a 1-minute granularity. Whenever possible, we strongly recommend that you scale on metrics with a 1-minute granularity to ensure a faster response to changes in system load.

To create a CloudWatch custom metric

1. Use the SQS [get-queue-attributes](#) command to get the number of messages waiting in the queue (ApproximateNumberOfMessages).

```
aws sqs get-queue-attributes --queue-url https://sqs.region.amazonaws.com/123456789/MyQueue \
--attribute-names ApproximateNumberOfMessages
```

2. Use the [describe-auto-scaling-groups](#) command to get the running capacity of the group, which is the number of instances in the InService lifecycle state. This command returns the instances of an Auto Scaling group along with their lifecycle state.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. Calculate the backlog per instance by dividing the approximate number of messages available for retrieval from the queue by the group's running capacity.
4. Publish the results at a 1-minute granularity as a CloudWatch custom metric.

Here is an example CLI [put-metric-data](#) command.

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance -- \
namespace MyNamespace \
--unit None --value 20 --
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

After your application is emitting the desired metric, the data is sent to CloudWatch. The metric is visible in the CloudWatch console. You can access it by logging into the AWS Management Console and navigating to the CloudWatch page. Then, view the metric by navigating to the metrics page or by searching for it using the search box. For information about viewing metrics, see [View available metrics](#) in the *Amazon CloudWatch User Guide*.

Step 2: Create a target tracking scaling policy

After publishing your custom metric, create a target tracking scaling policy with a customized metric specification.

To create a target tracking scaling policy

1. Use the following command to specify a target value for your scaling policy in a config.json file in your home directory.

For the TargetValue, calculate the acceptable backlog per instance metric and enter it here. To calculate this number, decide on a normal latency value and divide it by the average time that it takes to process a message.

```
$ cat ~/config.json
{
  "TargetValue": 100,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [
      {
        "Name": "MyOptionalMetricDimensionName",
        "Value": "MyOptionalMetricDimensionValue"
      }
    ],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

2. Use the [put-scaling-policy](#) command, along with the config.json file that you created in the previous step, to create your scaling policy.

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
\ --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
--target-tracking-configuration file://~/config.json
```

This creates two alarms: one for scaling out and one for scaling in. It also returns the Amazon Resource Name (ARN) of the policy that is registered with CloudWatch, which CloudWatch uses to invoke scaling whenever the metric threshold is in breach.

Step 3: Test your scaling policy

After your setup is complete, verify that your scaling policy is working. You can test it by increasing the number of messages in your SQS queue and then verifying that your Auto Scaling group has launched an additional EC2 instance. You can also test it by decreasing the number of messages in your SQS queue and then verifying that the Auto Scaling group has terminated an EC2 instance.

To test the scale-out function

1. Follow the steps in [Tutorial: Sending a message to an Amazon SQS queue](#) to add messages to your queue. Make sure that you have increased the number of messages in the queue so that the backlog per instance metric exceeds the target value.

It can take a few minutes for your changes to invoke the alarm.

2. Use the [describe-auto-scaling-groups](#) command to verify that the group has launched an instance.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

To test the scale-in function

1. Follow the steps in [Tutorial: Sending a message to an Amazon SQS queue](#) to remove messages from the queue. Make sure that you have decreased the number of messages in the queue so that the backlog per instance metric is below the target value.

It can take a few minutes for your changes to invoke the alarm.
2. Use the [describe-auto-scaling-groups](#) command to verify that the group has terminated an instance.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQS and instance scale-in protection

Messages that have not been processed at the time an instance is terminated are returned to the SQS queue where they can be processed by another instance that is still running. For applications where long running tasks are performed, you can optionally use instance scale-in protection to have control over which queue workers are terminated when your Auto Scaling group scales in.

The following pseudocode shows one way to protect long-running, queue-driven worker processes from scale-in termination.

```
while (true)
{
    SetInstanceProtection(False);
    Work = GetNextWorkUnit();
    SetInstanceProtection(True);
    ProcessWorkUnit(Work);
    SetInstanceProtection(False);
}
```

For more information, see [Use instance scale-in protection \(p. 240\)](#).

Verify a scaling activity for an Auto Scaling group

In the Amazon EC2 Auto Scaling section of the Amazon EC2 console, the **Activity history** for an Auto Scaling group lets you view the current status of a scaling activity that is currently in progress. When the scaling activity is finished, you can see whether it succeeds or not. This is particularly useful when you are creating Auto Scaling groups or you are adding scaling conditions to existing groups.

When you add a target tracking, step, or simple scaling policy to your Auto Scaling group, Amazon EC2 Auto Scaling immediately starts evaluating the policy against the metric. The metric alarm goes to ALARM state when the metric breaches the threshold for a specified number of evaluation periods. This means that a scaling policy could result in a scaling activity soon after it's created. After Amazon EC2 Auto Scaling adjusts the desired capacity in response to a scaling policy, you can verify the scaling activity in your account. If you want to receive email notification from Amazon EC2 Auto Scaling informing you about a scaling activity, follow the instructions in [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).

Tip

In the following procedure, you look at the **Activity history** and **Instances** sections for the Auto Scaling group. In both, the named columns should already be displayed. To display hidden columns or change the number of rows shown, choose the gear icon on the top right corner of each section to open the preferences modal, update the settings as needed, and choose **Confirm**.

To view the scaling activities for an Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances, or whether the scaling activity is still in progress.
4. (Optional) If you have a lot of scaling activities, you can choose the > icon at the top edge of the activity history to see the next page of scaling activities.
5. On the **Instance management** tab, under **Instances**, the **Lifecycle** column contains the state of your instances. After the instance starts and any lifecycle hooks have finished, its lifecycle state changes to **InService**. The **Health status** column shows the result of the EC2 instance health check on your instance.

To view the scaling activities for an Auto Scaling group (AWS CLI)

Use the following [describe-scaling-activities](#) command.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

The following is example output.

Scaling activities are ordered by start time. Activities still in progress are described first.

```
{  
  "Activities": [  
    {  
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",  
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-tracking-policy changing the desired capacity from 3 to 2. At 2020-02-11T18:34:31Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2020-02-11T18:34:31Z instance i-06c4794c2499af1df was selected for termination.",  
      "StartTime": "2020-02-11T18:34:31.268Z",  
      "EndTime": "2020-02-11T18:34:53Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\": \"subnet-5ea0c127\", \"Availability Zone\": \"us-west-2a\"...}",  
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:283179a2-f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"  
    },  
    ...  
  ]  
}
```

For a description of the fields in the output, see [Activity](#) in the *Amazon EC2 Auto Scaling API Reference*.

For help retrieving the scaling activities for a deleted group and for information about the types of errors that you may encounter and how to handle them, see [Troubleshoot Amazon EC2 Auto Scaling \(p. 360\)](#).

Disable a scaling policy for an Auto Scaling group

This topic describes how to temporarily disable a scaling policy so it won't initiate changes to the number of instances the Auto Scaling group contains. When you disable a scaling policy, the configuration details are preserved, so you can quickly re-enable the policy. This is easier than temporarily deleting a policy when you don't need it, and recreating it later.

When a scaling policy is disabled, the Auto Scaling group does not scale out or scale in for the metric alarms that are breached while the scaling policy is disabled. However, any scaling activities still in progress are not stopped.

Note that disabled scaling policies still count toward your quotas on the number of scaling policies that you can add to an Auto Scaling group.

To disable a scaling policy (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Automatic scaling** tab, under **Dynamic scaling policies**, select the check box in the top right corner of the desired scaling policy.
4. Scroll to the top of the **Dynamic scaling policies** section, and choose **Actions, Disable**.

When you are ready to re-enable the scaling policy, repeat these steps and then choose **Actions, Enable**. After you re-enable a scaling policy, your Auto Scaling group may immediately initiate a scaling action if there are any alarms currently in ALARM state.

To disable a scaling policy (AWS CLI)

Use the [put-scaling-policy](#) command with the `--no-enabled` option as follows. Specify all options in the command as you would specify them when creating the policy.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
--policy-name my-scaling-policy --policy-type TargetTrackingScaling \
--estimated-instance-warmup 360 \
--target-tracking-configuration '{ "TargetValue": 70, "PredefinedMetricSpecification": \
{ "PredefinedMetricType": "ASGAverageCPUUtilization" } }' \
--no-enabled
```

To re-enable a scaling policy (AWS CLI)

Use the [put-scaling-policy](#) command with the `--enabled` option as follows. Specify all options in the command as you would specify them when creating the policy.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
--policy-name my-scaling-policy --policy-type TargetTrackingScaling \
--estimated-instance-warmup 360 \
--target-tracking-configuration '{ "TargetValue": 70, "PredefinedMetricSpecification": \
{ "PredefinedMetricType": "ASGAverageCPUUtilization" } }' \
--enabled
```

To describe a scaling policy (AWS CLI)

Use the [describe-policies](#) command to verify the enabled status of a scaling policy.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \
```

```
--policy-names my-scaling-policy
```

The following is example output.

```
{  
  "ScalingPolicies": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "PolicyName": "my-scaling-policy",  
      "PolicyARN": "arn:aws:autoscaling:us-  
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-  
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",  
      "PolicyType": "TargetTrackingScaling",  
      "StepAdjustments": [],  
      "Alarms": [  
        {  
          "AlarmName": "TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-  
ae1199204502",  
          "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-  
ae1199204502"  
        },  
        {  
          "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-  
abe0-1cf8b9de6d6c",  
          "AlarmARN": "arn:aws:cloudwatch:us-  
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-  
abe0-1cf8b9de6d6c"  
        }  
      ],  
      "TargetTrackingConfiguration": {  
        "PredefinedMetricSpecification": {  
          "PredefinedMetricType": "ASGAverageCPUUtilization"  
        },  
        "TargetValue": 70.0,  
        "DisableScaleIn": false  
      },  
      "Enabled": true  
    }  
  ]  
}
```

Delete a scaling policy

After you no longer need a scaling policy, you can delete it. Depending on the type of scaling policy, you might also need to delete the CloudWatch alarms. Deleting a target tracking scaling policy also deletes any associated CloudWatch alarms. Deleting a step scaling policy or a simple scaling policy deletes the underlying alarm action, but it does not delete the CloudWatch alarm, even if it no longer has an associated action.

To delete a scaling policy (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Automatic scaling** tab, under **Dynamic scaling policies**, select the check box in the top right corner of the desired scaling policy.
4. Scroll to the top of the **Dynamic scaling policies** section, and choose **Actions, Delete**.

5. When prompted for confirmation, choose **Yes, Delete**.
6. (Optional) If you deleted a step scaling policy or a simple scaling policy, do the following to delete the CloudWatch alarm that was associated with the policy. You can skip these substeps to keep the alarm for future use.
 - a. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
 - b. On the navigation pane, choose **Alarms**.
 - c. Choose the alarm (for example, Step-Scaling-AlarmHigh-AddCapacity) and choose **Action, Delete**.
 - d. When prompted for confirmation, choose **Delete**.

To get the scaling policies for an Auto Scaling group (AWS CLI)

Before you delete a scaling policy, use the following [describe-policies](#) command to see what scaling policies were created for the Auto Scaling group. You can use the output when deleting the policy and the CloudWatch alarms.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

You can filter the results by the type of scaling policy using the `--query` parameter. This syntax for `query` works on Linux or macOS. On Windows, change the single quotes to double quotes.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
--query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

The following is example output.

```
[  
 {  
     "AutoScalingGroupName": "my-asg",  
     "PolicyName": "cpu40-target-tracking-scaling-policy",  
     "PolicyARN": "PolicyARN",  
     "PolicyType": "TargetTrackingScaling",  
     "StepAdjustments": [],  
     "Alarms": [  
         {  
             "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-  
asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",  
             "AlarmName": "TargetTracking-my-asg-AlarmHigh-  
fc0e4183-23ac-497e-9992-691c9980c38e"  
         },  
         {  
             "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-  
asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",  
             "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-  
bd9e-471a352ee1a2"  
         }  
     ],  
     "TargetTrackingConfiguration": {  
         "PredefinedMetricSpecification": {  
             "PredefinedMetricType": "ASGAverageCPUUtilization"  
         },  
         "TargetValue": 40.0,  
         "DisableScaleIn": false  
     },  
     "Enabled": true  
 }  
]
```

To delete your scaling policy (AWS CLI)

Use the following [delete-policy](#) command.

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \  
--policy-name cpu40-target-tracking-scaling-policy
```

To delete your CloudWatch alarm (AWS CLI)

For step and simple scaling policies, use the [delete-alarms](#) command to delete the CloudWatch alarm that was associated with the policy. You can skip this step to keep the alarm for future use. You can delete one or more alarms at a time. For example, use the following command to delete the Step-Scaling-AlarmHigh-AddCapacity and Step-Scaling-AlarmLow-RemoveCapacity alarms.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-  
AlarmLow-RemoveCapacity
```

Example scaling policies for the AWS Command Line Interface (AWS CLI)

You can create scaling policies for Amazon EC2 Auto Scaling through the AWS Management Console, AWS CLI, or SDKs.

The following examples show how you can create scaling policies for Amazon EC2 Auto Scaling with the AWS CLI [put-scaling-policy](#) command. For introductory exercises for creating scaling policies from the AWS CLI, see [Target tracking scaling policies \(p. 135\)](#) and [Step and simple scaling policies \(p. 141\)](#).

Example 1: To apply a target tracking scaling policy with a predefined metric specification

```
aws autoscaling put-scaling-policy --policy-name cpu40-target-tracking-scaling-policy \  
--auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
--target-tracking-configuration file://config.json  
{  
  "TargetValue": 40.0,  
  "PredefinedMetricSpecification": {  
    "PredefinedMetricType": "ASGAverageCPUUtilization"  
  }  
}
```

Example 2: To apply a target tracking scaling policy with a customized metric specification

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \  
--auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \  
--target-tracking-configuration file://config.json  
{  
  "TargetValue": 100.0,  
  "CustomizedMetricSpecification": {  
    "MetricName": "MyBacklogPerInstance",  
    "Namespace": "MyNamespace",  
    "Dimensions": [{  
      "Name": "MyOptionalMetricDimensionName",  
      "Value": "MyOptionalMetricDimensionValue"  
    }],  
    "Statistic": "Average",  
    "Unit": "None"  
  }  
}
```

Example 3: To apply a target tracking scaling policy for scale out only

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy \
--auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
--target-tracking-configuration file://config.json
{
    "TargetValue": 1000.0,
    "PredefinedMetricSpecification": {
        "PredefinedMetricType": "ALBRequestCountPerTarget",
        "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-group/943f017f100becff"
    },
    "DisableScaleIn": true
}
```

Example 4: To apply a step scaling policy for scale out

```
aws autoscaling put-scaling-policy \
--auto-scaling-group-name my-asg \
--policy-name my-step-scale-out-policy \
--policy-type StepScaling \
--adjustment-type PercentChangeInCapacity \
--metric-aggregation-type Average \
--step-adjustments
    MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \
    MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
        MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
--min-adjustment-magnitude 1
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 5: To apply a step scaling policy for scale in

```
aws autoscaling put-scaling-policy \
--auto-scaling-group-name my-asg \
--policy-name my-step-scale-in-policy \
--policy-type StepScaling \
--adjustment-type ChangeInCapacity \
--step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 6: To apply a simple scaling policy for scale out

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
--auto-scaling-group-name my-asg --scaling-adjustment 30 \
--adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 7: To apply a simple scaling policy for scale in

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
--auto-scaling-group-name my-asg --scaling-adjustment -1 \
--adjustment-type ChangeInCapacity --cooldown 180
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Predictive scaling for Amazon EC2 Auto Scaling

Use predictive scaling to increase the number of EC2 instances in your Auto Scaling group in advance of daily and weekly patterns in traffic flows.

Predictive scaling is well suited for situations where you have:

- Cyclical traffic, such as high use of resources during regular business hours and low use of resources during evenings and weekends
- Recurring on-and-off workload patterns, such as batch processing, testing, or periodic data analysis
- Applications that take a long time to initialize, causing a noticeable latency impact on application performance during scale-out events

In general, if you have regular patterns of traffic increases and applications that take a long time to initialize, you should consider using predictive scaling. Predictive scaling can help you scale faster by launching capacity in advance of forecasted load, compared to using only dynamic scaling, which is reactive in nature. Predictive scaling can also potentially save you money on your EC2 bill by helping you avoid the need to overprovision capacity.

For example, consider an application that has high usage during business hours and low usage overnight. At the start of each business day, predictive scaling can add capacity before the first influx of traffic. This helps your application maintain high availability and performance when going from a period of lower utilization to a period of higher utilization. You don't have to wait for dynamic scaling to react to changing traffic. You also don't have to spend time reviewing your application's load patterns and trying to schedule the right amount of capacity using scheduled scaling.

Use the AWS Management Console, the AWS CLI, or one of the SDKs to add a predictive scaling policy to any Auto Scaling group.

Contents

- [How predictive scaling works \(p. 168\)](#)
- [Best practices \(p. 169\)](#)
- [Create a predictive scaling policy \(console\) \(p. 169\)](#)
- [Create a predictive scaling policy \(AWS CLI\) \(p. 172\)](#)
- [Limitations \(p. 174\)](#)
- [Supported Regions \(p. 174\)](#)
- [Explore your data and forecast \(p. 175\)](#)
- [Override forecast values using scheduled actions \(p. 179\)](#)
- [Advanced predictive scaling policy configurations using custom metrics \(p. 183\)](#)

How predictive scaling works

Predictive scaling uses machine learning to predict capacity requirements based on historical data from CloudWatch. The machine learning algorithm consumes the available historical data and calculates capacity that best fits the historical load pattern, and then continuously learns based on new data to make future forecasts more accurate.

To use predictive scaling, you first create a scaling policy with a pair of metrics and a target utilization. Forecast creation starts immediately after you create your policy if there is at least 24 hours of historical data. Predictive scaling finds patterns in CloudWatch metric data from the previous 14 days to create an hourly forecast for the next 48 hours. Forecast data is updated daily based on the most recent CloudWatch metric data.

You can configure predictive scaling in *forecast only* mode so that you can evaluate the forecast before predictive scaling starts actively scaling capacity. You can then view the forecast and recent metric data from CloudWatch in graph form from the Amazon EC2 Auto Scaling console. You can also access forecast data by using the AWS CLI or one of the SDKs.

When you are ready to start scaling with predictive scaling, switch the policy from *forecast only* mode to *forecast and scale* mode. After you switch to *forecast and scale* mode, your Auto Scaling group starts scaling based on the forecast.

Using the forecast, Amazon EC2 Auto Scaling scales the number of instances at the beginning of each hour:

- If actual capacity is less than the predicted capacity, Amazon EC2 Auto Scaling scales out your Auto Scaling group so that its desired capacity is equal to the predicted capacity.
- If actual capacity is greater than the predicted capacity, Amazon EC2 Auto Scaling doesn't scale in capacity.
- The values that you set for the minimum and maximum capacity of the Auto Scaling group are adhered to if the predicted capacity is outside of this range.

Best practices

- Confirm whether predictive scaling is suitable for your workload. A workload is a good fit for predictive scaling if it exhibits recurring load patterns that are specific to the day of the week or the time of day. To check this, configure predictive scaling policies in *forecast only* mode.
- Evaluate the forecast and its accuracy before allowing predictive scaling to actively scale your application. Predictive scaling needs at least 24 hours of historical data to start forecasting. However, forecasts are more effective if historical data spans the full two weeks. If you update your application by creating a new Auto Scaling group and deleting the old one, then your new Auto Scaling group needs 24 hours of historical load data before predictive scaling can start generating forecasts again. In this case, you might have to wait a few days for a more accurate forecast.
- Create multiple predictive scaling policies in *forecast only* mode to test the potential effects of different metrics. You can create multiple predictive scaling policies for each Auto Scaling group, but only one of the policies can be used for active scaling.
- If you choose a custom metric pair, you can define a different combination of load metric and scaling metric. To avoid issues, make sure that the load metric you choose represents the full load on your application.
- Use predictive scaling with dynamic scaling. Dynamic scaling is used to automatically scale capacity in response to real-time changes in resource utilization. Using it with predictive scaling helps you follow the demand curve for your application closely, scaling in during periods of low traffic and scaling out when traffic is higher than expected. When multiple scaling policies are active, each policy determines the desired capacity independently, and the desired capacity is set to the maximum of those. For example, if 10 instances are required to stay at the target utilization in a target tracking scaling policy, and 8 instances are required to stay at the target utilization in a predictive scaling policy, then the group's desired capacity is set to 10.

Create a predictive scaling policy (console)

You can create, view, and delete your predictive scaling policies with the Amazon EC2 Auto Scaling console.

[Create a predictive scaling policy in the console \(predefined metrics\)](#)

Use the following procedure to create a predictive scaling policy using predefined metrics (CPU, network I/O, or Application Load Balancer request count per target). The easiest way to create a predictive scaling

policy is to use predefined metrics. If you prefer to use custom metrics instead, see [Create a predictive scaling policy in the console \(custom metrics\) \(p. 171\)](#).

If the Auto Scaling group is new, it must provide at least 24 hours of data before Amazon EC2 Auto Scaling can generate a forecast for it.

To create a predictive scaling policy

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up at the bottom of the page.

3. On the **Automatic scaling** tab, in **Scaling policies**, choose **Create predictive scaling policy**.
4. Enter a name for the policy.
5. Turn on **Scale based on forecast** to give Amazon EC2 Auto Scaling permission to start scaling right away.

To keep the policy in *forecast only* mode, keep **Scale based on forecast** turned off.

6. For **Metrics**, choose your metrics from the list of options. Options include **CPU**, **Network In**, **Network Out**, **Application Load Balancer request count**, and **Custom metric pair**.

If you chose **Application Load Balancer request count per target**, then choose a target group in **Target group**. **Application Load Balancer request count per target** is only supported if you have attached an Application Load Balancer target group to your Auto Scaling group.

If you chose **Custom metric pair**, choose individual metrics from the drop-down lists for **Load metric** and **Scaling metric**.

7. For **Target utilization**, enter the target value that Amazon EC2 Auto Scaling should maintain. Amazon EC2 Auto Scaling scales out your capacity until the average utilization is at the target utilization, or until it reaches the maximum number of instances you specified.

If your scaling metric is...	Then the target utilization represents...
CPU	The percentage of CPU that each instance should ideally use.
Network In	The average number of bytes per minute that each instance should ideally receive.
Network Out	The average number of bytes per minute that each instance should ideally send out.
Application Load Balancer request count per target	The average number of requests per minute that each instance should ideally receive.

8. (Optional) For **Pre-launch instances**, choose how far in advance you want your instances launched before the forecast calls for the load to increase.
9. (Optional) For **Max capacity behavior**, choose whether to let Amazon EC2 Auto Scaling scale out higher than the group's maximum capacity when predicted capacity exceeds the defined maximum. Turning on this setting lets scale out occur during periods when your traffic is forecasted to be at its highest.
10. (Optional) For **Buffer maximum capacity above the forecasted capacity**, choose how much additional capacity to use when the predicted capacity is close to or exceeds the maximum capacity. The value is specified as a percentage relative to the predicted capacity. For example, if the buffer is 10, this means a 10 percent buffer. Therefore, if the predicted capacity is 50 and the maximum capacity is 40, the effective maximum capacity is 55.

If set to 0, Amazon EC2 Auto Scaling might scale capacity higher than the maximum capacity to equal but not exceed predicted capacity.

11. Choose **Create predictive scaling policy**.

Create a predictive scaling policy in the console (custom metrics)

Use the following procedure to create a predictive scaling policy using custom metrics. Custom metrics can include other metrics provided by CloudWatch or metrics that you publish to CloudWatch. To use CPU, network I/O, or Application Load Balancer request count per target, see [Create a predictive scaling policy in the console \(predefined metrics\) \(p. 169\)](#).

To create a predictive scaling policy using custom metrics, you must do the following:

- You must provide the raw queries that let Amazon EC2 Auto Scaling interact with the metrics in CloudWatch. For more information, see [Advanced predictive scaling policy configurations using custom metrics \(p. 183\)](#). To be sure that Amazon EC2 Auto Scaling can extract the metric data from CloudWatch, confirm that each query is returning data points. Confirm this by using the CloudWatch console or the CloudWatch [GetMetricData](#) API operation.

Note

We provide sample JSON payloads in the JSON editor in the Amazon EC2 Auto Scaling console. These examples give you a reference for the key-value pairs that are required to add other CloudWatch metrics provided by AWS or metrics that you previously published to CloudWatch. You can use them as a starting point, then customize them for your needs.

- If you use any metric math, you must manually construct the JSON to fit your unique scenario. For more information, see [Use metric math expressions \(p. 185\)](#). Before using metric math in your policy, confirm that metric queries based on metric math expressions are valid and return a single time series. Confirm this by using the CloudWatch console or the CloudWatch [GetMetricData](#) API operation.

If you make an error in a query by providing incorrect data, such as the wrong Auto Scaling group name, the forecast won't have any data. For troubleshooting custom metric issues, see [Considerations and troubleshooting \(p. 189\)](#).

To create a predictive scaling policy

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up at the bottom of the page.

3. On the **Automatic scaling** tab, in **Scaling policies**, choose **Create predictive scaling policy**.
4. Enter a name for the policy.
5. Turn on **Scale based on forecast** to give Amazon EC2 Auto Scaling permission to start scaling right away.

To keep the policy in *forecast only* mode, keep **Scale based on forecast** turned off.

6. For **Metrics**, choose **Custom metric pair**.
 - a. For **Load metric**, choose **Custom CloudWatch metric** to use a custom metric. Construct the JSON payload that contains the load metric definition for the policy and paste it into the JSON editor box, replacing what is already in the box.
 - b. For **Scaling metric**, choose **Custom CloudWatch metric** to use a custom metric. Construct the JSON payload that contains the scaling metric definition for the policy and paste it into the JSON editor box, replacing what is already in the box.

- c. (Optional) To add a custom capacity metric, select the check box for **Add custom capacity metric**. Construct the JSON payload that contains the capacity metric definition for the policy and paste it into the JSON editor box, replacing what is already in the box.

You only need to enable this option to create a new time series for capacity if your capacity metric data spans multiple Auto Scaling groups. In this case, you must use metric math to aggregate the data into a single time series.

7. For **Target utilization**, enter the target value that Amazon EC2 Auto Scaling should maintain. Amazon EC2 Auto Scaling scales out your capacity until the average utilization is at the target utilization, or until it reaches the maximum number of instances you specified.
 8. (Optional) For **Pre-launch instances**, choose how far in advance you want your instances launched before the forecast calls for the load to increase.
 9. (Optional) For **Max capacity behavior**, choose whether to let Amazon EC2 Auto Scaling scale out higher than the group's maximum capacity when predicted capacity exceeds the defined maximum. Turning on this setting lets scale out occur during periods when your traffic is forecasted to be at its highest.
 10. (Optional) For **Buffer maximum capacity above the forecasted capacity**, choose how much additional capacity to use when the predicted capacity is close to or exceeds the maximum capacity. The value is specified as a percentage relative to the predicted capacity. For example, if the buffer is 10, this means a 10 percent buffer. Therefore, if the predicted capacity is 50 and the maximum capacity is 40, the effective maximum capacity is 55.
- If set to 0, Amazon EC2 Auto Scaling might scale capacity higher than the maximum capacity to equal but not exceed predicted capacity.
11. Choose **Create predictive scaling policy**.

Create a predictive scaling policy (AWS CLI)

Use the AWS CLI as follows to configure predictive scaling policies for your Auto Scaling group. For more information about the CloudWatch metrics you can specify for a predictive scaling policy, see [PredictiveScalingMetricSpecification](#) in the *Amazon EC2 Auto Scaling API Reference*.

Example 1: A predictive scaling policy that creates forecasts but doesn't scale

The following example policy shows a complete policy configuration that uses CPU utilization metrics for predictive scaling with a target utilization of 40. ForecastOnly mode is used by default, unless you explicitly specify which mode to use. Save this configuration in a file named config.json.

```
{  
  "MetricSpecifications": [  
    {  
      "TargetValue": 40,  
      "PredefinedMetricPairSpecification": {  
        "PredefinedMetricType": "ASGCPUUtilization"  
      }  
    }  
  ]  
}
```

To create the policy from the command line, run the [put-scaling-policy](#) command with the configuration file specified, as demonstrated in the following example.

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \  
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
```

```
--predictive-scaling-configuration file://config.json
```

If successful, this command returns the policy's Amazon Resource Name (ARN).

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-  
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-policy",  
  "Alarms": []  
}
```

Example 2: A predictive scaling policy that forecasts and scales

For a policy that allows Amazon EC2 Auto Scaling to forecast and scale, add the property `Mode` with a value of `ForecastAndScale`. The following example shows a policy configuration that uses Application Load Balancer request count metrics. The target utilization is `1000`, and predictive scaling is set to `ForecastAndScale` mode.

```
{  
  "MetricSpecifications": [  
    {  
      "TargetValue": 1000,  
      "PredefinedMetricPairSpecification": {  
        "PredefinedMetricType": "ALBRequestCount",  
        "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-  
group/943f017f100becff"  
      }  
    },  
    {"Mode": "ForecastAndScale"}  
  ]  
}
```

To create this policy, run the [put-scaling-policy](#) command with the configuration file specified, as demonstrated in the following example.

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \  
--auto-scaling-group-name my-asg --policy-type PredictiveScaling \  
--predictive-scaling-configuration file://config.json
```

If successful, this command returns the policy's Amazon Resource Name (ARN).

```
{  
  "PolicyARN": "arn:aws:autoscaling:region:account-  
id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-  
asg:policyName/alb1000-predictive-scaling-policy",  
  "Alarms": []  
}
```

Example 3: A predictive scaling policy that can scale higher than maximum capacity

The following example shows how to create a policy that can scale higher than the group's maximum size limit when you need it to handle a higher than normal load. By default, Amazon EC2 Auto Scaling doesn't scale your EC2 capacity higher than your defined maximum capacity. However, it might be helpful to let it scale higher with slightly more capacity to avoid performance or availability issues.

To provide room for Amazon EC2 Auto Scaling to provision additional capacity when the capacity is predicted to be at or very close to your group's maximum size, specify the `MaxCapacityBreachBehavior` and `MaxCapacityBuffer` properties, as shown in the following

example. You must specify `MaxCapacityBreachBehavior` with a value of `IncreaseMaxCapacity`. The maximum number of instances that your group can have depends on the value of `MaxCapacityBuffer`.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 70,
      "PredefinedMetricPairSpecification": {
        "PredefinedMetricType": "ASGCPUUtilization"
      }
    }
  ],
  "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
  "MaxCapacityBuffer": 10
}
```

In this example, the policy is configured to use a 10 percent buffer ("`MaxCapacityBuffer`": 10), so if the predicted capacity is 50 and the maximum capacity is 40, then the effective maximum capacity is 55. A policy that can scale capacity higher than the maximum capacity to equal but not exceed predicted capacity would have a buffer of 0 ("`MaxCapacityBuffer`": 0).

To create this policy, run the [put-scaling-policy](#) command with the configuration file specified, as demonstrated in the following example.

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \
--auto-scaling-group-name my-asg --policy-type PredictiveScaling \
--predictive-scaling-configuration file://config.json
```

If successful, this command returns the policy's Amazon Resource Name (ARN).

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-policy",
  "Alarms": []
}
```

Limitations

- Predictive scaling requires 24 hours of metric history before it can generate forecasts.
- A core assumption of predictive scaling is that the Auto Scaling group is homogenous and all instances are of equal capacity. If this isn't true for your group, forecasted capacity can be inaccurate. Therefore, use caution when creating predictive scaling policies for [mixed instances groups \(p. 52\)](#), because instances of different types can be provisioned that are of unequal capacity. Following are some examples where the forecasted capacity will be inaccurate:
 - Your predictive scaling policy is based on CPU utilization, but the number of vCPUs on each Auto Scaling instance varies between instance types.
 - Your predictive scaling policy is based on network in or network out, but the network bandwidth throughput for each Auto Scaling instance varies between instance types. For example, the M5 and M5n instance types are similar, but the M5n instance type delivers significantly higher network throughput.

Supported Regions

Amazon EC2 Auto Scaling supports predictive scaling policies in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), US West (N. California), Africa (Cape Town), Canada (Central),

EU (Frankfurt), EU (Ireland), EU (London), EU (Milan), EU (Paris), EU (Stockholm), Asia Pacific (Hong Kong), Asia Pacific (Jakarta), Asia Pacific (Mumbai), Asia Pacific (Osaka), Asia Pacific (Tokyo), Asia Pacific (Singapore), Asia Pacific (Seoul), Asia Pacific (Sydney), Middle East (Bahrain), South America (Sao Paulo), China (Beijing), China (Ningxia), and AWS GovCloud (US-West).

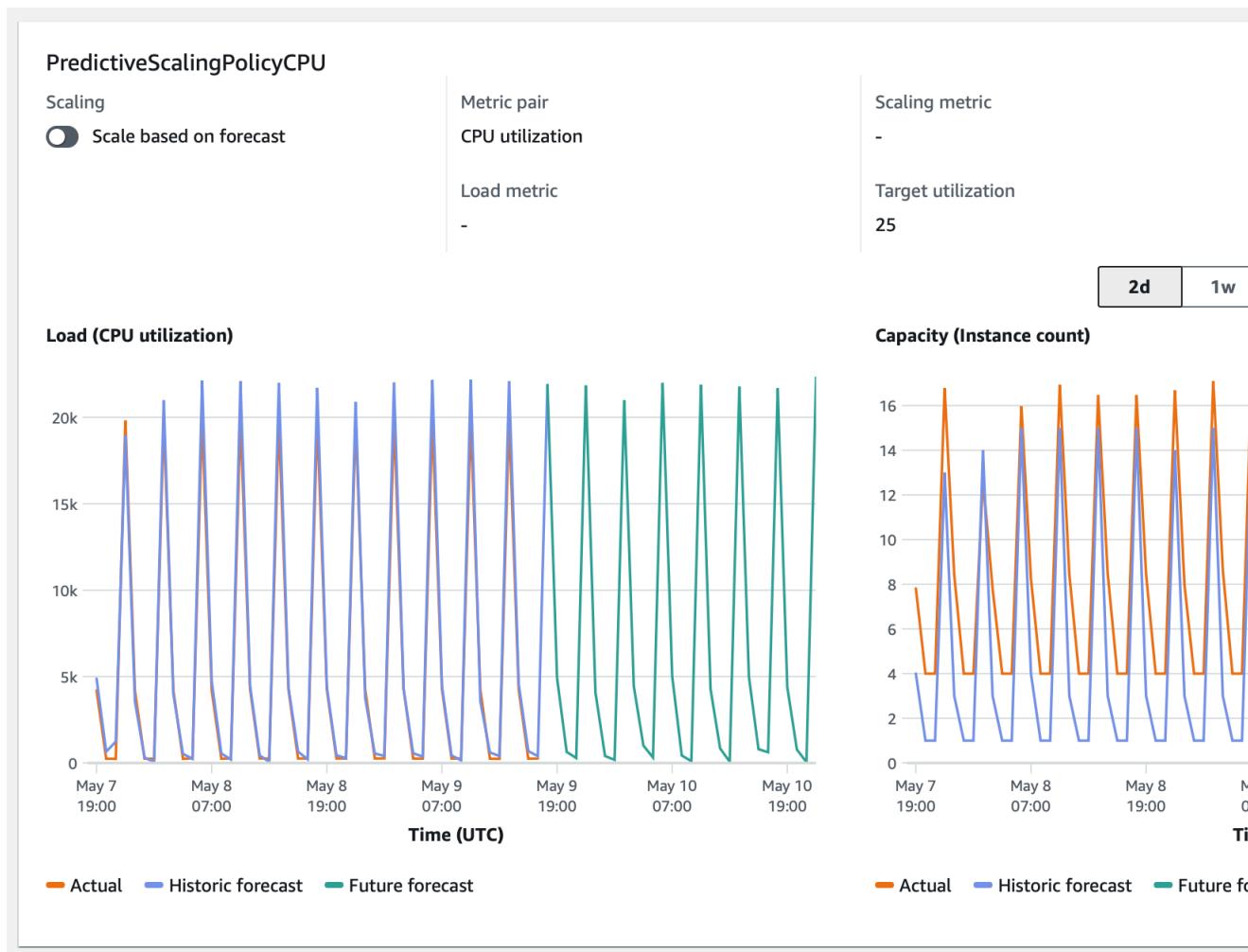
Explore your data and forecast

After the forecast is created, you can view graphs showing historical data from the last eight weeks and the forecast for the next two days. The graphs become available shortly after the policy is created.

To view the forecast and its history using the Amazon EC2 Auto Scaling console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.
A split pane opens up in the bottom of the page.
3. In the lower pane, choose the **Automatic scaling** tab.

Each graph displays forecast values against actual values and a particular type of data. The **Load** graph shows load forecast and actual values for the load metric you choose. The **Capacity** graph shows the number of instances that are forecasted based on your target utilization and the actual number of instances launched. Various colors show actual metric data points and past and future forecast values. The orange line shows the actual metric data points. The green line shows the forecast that was generated for the future forecast period. The blue line shows the forecast for past periods.



You can adjust the time range for past data by choosing your preferred value in the top right of the graph: **2 days, 1 week, 2 weeks, 4 weeks, 6 weeks, or 8 weeks**. Each point on the graph represents one hour of data. When hovering over a data point, the tooltip shows the value for a particular point in time, in UTC.

To enlarge the graph pane, choose the expand icon in the top right of the graph. To revert back to the default view, choose the icon again.

You can also use the AWS CLI command **get-predictive-scaling-forecast** to get forecast data. The data returned by this call can help you identify time periods when you might want to override the forecast. For more information, see [Override forecast values using scheduled actions \(p. 179\)](#).

Note

We recommend that you enable Auto Scaling group metrics. If these metrics are not enabled, actual capacity data will be missing from the capacity forecast graph. There is no cost for enabling these metrics. For more information, see [Enable Auto Scaling group metrics \(console\) \(p. 264\)](#).

Important

If the Auto Scaling group is new, allow 24 hours for Amazon EC2 Auto Scaling to create the first forecast.

Monitor predictive scaling metrics with CloudWatch

Depending on your needs, you might prefer to access monitoring data for predictive scaling from Amazon CloudWatch instead of the Amazon EC2 Auto Scaling console. After you create a predictive scaling policy, the policy collects data that is used to forecast your future load and capacity. After this data is collected, it is automatically stored in CloudWatch at regular intervals. Then, you can use CloudWatch to visualize how well the policy performs over time. You can also create CloudWatch alarms to notify you when performance indicators change beyond the limits that you define in CloudWatch.

Topics

- [Visualize historical forecast data \(p. 177\)](#)
- [Create accuracy metrics using metric math \(p. 177\)](#)

Visualize historical forecast data

You can view the load and capacity forecast data for a predictive scaling policy in CloudWatch. This can be useful when visualizing forecasts against other CloudWatch metrics in a single graph. It can also help when viewing a broader time range so that you can see trends over time. You can access up to 15 months of historical metrics to get a better perspective on how your policy is performing.

For more information, see [Predictive scaling metrics and dimensions \(p. 264\)](#).

To view historical forecast data using the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics** and then **All metrics**.
3. Choose the **Auto Scaling** metric namespace.
4. Choose one of the following options to view either the load forecast or capacity forecast metrics:
 - **Predictive Scaling Load Forecasts**
 - **Predictive Scaling Capacity Forecasts**
5. In the search field, enter the name of the predictive scaling policy or the name of the Auto Scaling group, and then press Enter to filter the results.
6. To graph a metric, select the check box next to the metric. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**. For more information, see [Graphing a metric](#) in the *Amazon CloudWatch User Guide*.
7. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different statistic. Although you can choose any statistic for each metric, not all statistics are useful for **PredictiveScalingLoadForecast** and **PredictiveScalingCapacityForecast** metrics. For example, the **Average**, **Minimum**, and **Maximum** statistics are useful, but the **Sum** statistic is not.
8. To add another metric to the graph, under **Browse**, choose **All**, find the specific metric, and then select the check box next to it. You can add up to 10 metrics.

For example, to add the actual values for CPU utilization to the graph, choose the **EC2** namespace and then choose **By Auto Scaling Group**. Then, select the check box for the **CPUUtilization** metric and the specific Auto Scaling group.

9. (Optional) To add the graph to a CloudWatch dashboard, choose **Actions, Add to dashboard**.

Create accuracy metrics using metric math

With metric math, you can query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics. You can visualize the resulting time series on the CloudWatch console

and add them to dashboards. For more information about metric math, see [Using metric math](#) in the [Amazon CloudWatch User Guide](#).

Using metric math, you can graph the data that Amazon EC2 Auto Scaling generates for predictive scaling in different ways. This helps you monitor policy performance over time, and helps you understand whether your combination of metrics can be improved.

For example, you can use a metric math expression to monitor the [mean absolute percentage error](#) (MAPE). The MAPE metric helps monitor the difference between the forecasted values and the actual values observed during a given forecast window. Changes in the value of MAPE can indicate whether the policy's performance is degrading over time as the nature of your application changes. An increase in MAPE signals a wider gap between the forecasted values and the actual values.

Example: Metric math expression

To get started with this type of graph, you can create a metric math expression like the one shown in the following example.

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "m2",
      "Label": "ForecastedLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/AutoScaling",
          "MetricName": "PredictiveScalingLoadForecast",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            },
            {
              "Name": "PolicyName",
              "Value": "my-predictive-scaling-policy"
            }
          ],
          "Name": "PairIndex",
          "Value": "1"
        }
      }
    }
  ]
}
```

```
        "Value": "0"
    }
]
},
"Period": 3600,
"Stat": "Average"
},
"ReturnData": false
]
}
```

Instead of a single metric, there is an array of metric data query structures for `MetricDataQueries`. Each item in `MetricDataQueries` gets a metric or performs a math expression. The first item, `e1`, is the math expression. The designated expression sets the `ReturnData` parameter to `true`, which ultimately produces a single time series. For all other metrics, the `ReturnData` value is `false`.

In the example, the designated expression uses the actual and forecasted values as input and returns the new metric (MAPE). `m1` is the CloudWatch metric that contains the actual load values (assuming CPU utilization is the load metric that was originally specified for the policy named `my-predictive-scaling-policy`). `m2` is the CloudWatch metric that contains the forecasted load values. The math syntax for the MAPE metric is as follows:

Average of (abs ((Actual - Forecast)/(Actual)))

Visualize your accuracy metrics and set alarms

To visualize the accuracy metric data, select the **Metrics** tab in the CloudWatch console. You can graph the data from there. For more information, see [Adding a math expression to a CloudWatch graph](#) in the *Amazon CloudWatch User Guide*.

You can also set an alarm on a metric that you're monitoring from the **Metrics** section. While on the **Graphed metrics** tab, select the **Create alarm** icon under the **Actions** column. The **Create alarm** icon is represented as a small bell. For more information, see [Creating a CloudWatch alarm based on a metric math expression](#) in the *Amazon CloudWatch User Guide*. For information about receiving alerts with Amazon SNS, see [Setting up Amazon SNS notifications](#) in the *Amazon CloudWatch User Guide*.

Alternatively, you can use [GetMetricData](#) and [PutMetricAlarm](#) to perform calculations using metric math and create alarms based on the output.

Override forecast values using scheduled actions

Sometimes, you might have additional information about your future application requirements that the forecast calculation is unable to take into account. For example, forecast calculations might underestimate the capacity needed for an upcoming marketing event. You can use scheduled actions to temporarily override the forecast during future time periods. The scheduled actions can run on a recurring basis, or at a specific date and time when there are one-time demand fluctuations.

For example, you can create a scheduled action with a higher minimum capacity than what is forecasted. At runtime, Amazon EC2 Auto Scaling updates the minimum capacity of your Auto Scaling group. Because predictive scaling optimizes for capacity, a scheduled action with a minimum capacity that is higher than the forecast values is honored. This prevents capacity from being less than expected. To stop overriding the forecast, use a second scheduled action to return the minimum capacity to its original setting.

The following procedure outlines the steps for overriding the forecast during future time periods.

Contents

- [Step 1: \(Optional\) Analyze time series data \(p. 180\)](#)

- [Step 2: Create two scheduled actions \(p. 181\)](#)

Step 1: (Optional) Analyze time series data

Start by analyzing the forecast time series data. This is an optional step, but it is helpful if you want to understand the details of the forecast.

1. Retrieve the forecast

After the forecast is created, you can query for a specific time period in the forecast. The goal of the query is to get a complete view of the time series data for a specific time period.

Your query can include up to two days of future forecast data. If you have been using predictive scaling for a while, you can also access your past forecast data. However, the maximum time duration between the start and end time is 30 days.

To get the forecast using the [get-predictive-scaling-forecast](#) AWS CLI command, provide the following parameters in the command:

- Enter the name of the Auto Scaling group in the `--auto-scaling-group-name` parameter.
- Enter the name of the policy in the `--policy-name` parameter.
- Enter the start time in the `--start-time` parameter to return only forecast data for after or at the specified time.
- Enter the end time in the `--end-time` parameter to return only forecast data for before the specified time.

```
aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \
  --policy-name cpu40-predictive-scaling-policy \
  --start-time "2021-05-19T17:00:00Z" \
  --end-time "2021-05-19T23:00:00Z"
```

If successful, the command returns data similar to the following example.

```
{  
  "LoadForecast": [  
    {  
      "Timestamps": [  
        "2021-05-19T17:00:00+00:00",  
        "2021-05-19T18:00:00+00:00",  
        "2021-05-19T19:00:00+00:00",  
        "2021-05-19T20:00:00+00:00",  
        "2021-05-19T21:00:00+00:00",  
        "2021-05-19T22:00:00+00:00",  
        "2021-05-19T23:00:00+00:00"  
      ],  
      "Values": [  
        153.0655799339254,  
        128.8288551285919,  
        107.1179447150675,  
        197.3601844551528,  
        626.4039934516954,  
        596.9441277518481,  
        677.9675713779869  
      ],  
      "MetricSpecification": {  
        "TargetValue": 40.0,  
        "PredefinedMetricPairSpecification": {  
          "PredefinedMetricType": "ASGCPUUtilization"  
        }  
      }  
    }  
  ]  
}
```

```
        }
    }
],
"CapacityForecast": {
    "Timestamps": [
        "2021-05-19T17:00:00+00:00",
        "2021-05-19T18:00:00+00:00",
        "2021-05-19T19:00:00+00:00",
        "2021-05-19T20:00:00+00:00",
        "2021-05-19T21:00:00+00:00",
        "2021-05-19T22:00:00+00:00",
        "2021-05-19T23:00:00+00:00"
    ],
    "Values": [
        2.0,
        2.0,
        2.0,
        2.0,
        4.0,
        4.0,
        4.0
    ]
},
"UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

The response includes two forecasts: `LoadForecast` and `CapacityForecast`. `LoadForecast` shows the hourly load forecast. `CapacityForecast` shows forecast values for the capacity that is needed on an hourly basis to handle the forecasted load while maintaining a `TargetValue` of 40.0 (40% average CPU utilization).

2. Identify the target time period

Identify the hour or hours when the one-time demand fluctuation should take place. Remember that dates and times shown in the forecast are in UTC.

Step 2: Create two scheduled actions

Next, create two scheduled actions for a specific time period when your application will have a higher than forecasted load. For example, if you have a marketing event that will drive traffic to your site for a limited period of time, you can schedule a one-time action to update the minimum capacity when it starts. Then, schedule another action to return the minimum capacity to the original setting when the event ends.

To create two scheduled actions for one-time events (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.
A split pane opens up in the bottom of the page.
3. On the **Automatic scaling** tab, in **Scheduled actions**, choose **Create scheduled action**.
4. Fill in the following scheduled action settings:
 - a. Enter a **Name** for the scheduled action.
 - b. For **Min**, enter the new minimum capacity for your Auto Scaling group. The **Min** must be less than or equal to the maximum size of the group. If your value for **Min** is greater than group's maximum size, you must update **Max**.

- c. For **Recurrence**, choose **Once**.
 - d. For **Time zone**, choose a time zone. If no time zone is chosen, ETC/UTC is used by default.
 - e. Define a **Specific start time**.
 5. Choose **Create**.
- The console displays the scheduled actions for the Auto Scaling group.
6. Configure a second scheduled action to return the minimum capacity to the original setting at the end of the event. Predictive scaling can scale capacity only when the value you set for **Min** is lower than the forecast values.

To create two scheduled actions for one-time events (AWS CLI)

To use the AWS CLI to create the scheduled actions, use the [put-scheduled-update-group-action](#) command.

For example, let's define a schedule that maintains a minimum capacity of three instances on May 19 at 5:00 PM for eight hours. The following commands show how to implement this scenario.

The first [put-scheduled-update-group-action](#) command instructs Amazon EC2 Auto Scaling to update the minimum capacity of the specified Auto Scaling group at 5:00 PM UTC on May 19, 2021.

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

The second command instructs Amazon EC2 Auto Scaling to set the group's minimum capacity to one at 1:00 AM UTC on May 20, 2021.

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

After you add these scheduled actions to the Auto Scaling group, Amazon EC2 Auto Scaling does the following:

- At 5:00 PM UTC on May 19, 2021, the first scheduled action runs. If the group currently has fewer than three instances, the group scales out to three instances. During this time and for the next eight hours, Amazon EC2 Auto Scaling can continue to scale out if the predicted capacity is higher than the actual capacity or if there is a dynamic scaling policy in effect.
- At 1:00 AM UTC on May 20, 2021, the second scheduled action runs. This returns the minimum capacity to its original setting at the end of the event.

Scaling based on recurring schedules

To override the forecast for the same time period every week, create two scheduled actions and provide the time and date logic using a cron expression.

The cron expression format consists of five fields separated by spaces: [Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]. Fields can contain any allowed values, including special characters.

For example, the following cron expression runs the action every Tuesday at 6:30 AM. The asterisk is used as a wildcard to match all values for a field.

```
30 6 * * 2
```

See also

For more information about how to create, list, edit, and delete scheduled actions, see [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#).

Advanced predictive scaling policy configurations using custom metrics

In a predictive scaling policy, you can use predefined or custom metrics. Custom metrics are useful when the predefined metrics (CPU, network I/O, and Application Load Balancer request count) do not sufficiently describe your application load.

When creating a predictive scaling policy with custom metrics, you can specify other CloudWatch metrics provided by AWS, or you can specify metrics that you define and publish yourself. You can also use metric math to aggregate and transform existing metrics into a new time series that AWS doesn't automatically track. When you combine values in your data, for example, by calculating new sums or averages, it's called *aggregating*. The resulting data is called an *aggregate*.

The following section contains best practices and examples of how to construct the JSON structure for the policy.

Contents

- [Best practices \(p. 183\)](#)
- [Prerequisites \(p. 184\)](#)
- [Constructing the JSON for custom metrics \(p. 184\)](#)
- [Considerations and troubleshooting \(p. 189\)](#)
- [Limitations \(p. 190\)](#)

Best practices

The following best practices can help you use custom metrics more effectively:

- For the load metric specification, the most useful metric is a metric that represents the load on an Auto Scaling group as a whole, regardless of the group's capacity.
- For the scaling metric specification, the most useful metric to scale by is an average throughput or utilization per instance metric.
- The scaling metric must be inversely proportional to capacity. That is, if the number of instances in the Auto Scaling group increases, the scaling metric should decrease by roughly the same proportion. To ensure that predictive scaling behaves as expected, the load metric and the scaling metric must also correlate strongly with each other.
- The target utilization must match the type of scaling metric. For a policy configuration that uses CPU utilization, this is a target percentage. For a policy configuration that uses throughput, such as the number of requests or messages, this is the target number of requests or messages per instance during any one-minute interval.
- If these recommendations are not followed, the forecasted future values of the time series will probably be incorrect. To validate that the data is correct, you can view the forecasted values in the Amazon EC2 Auto Scaling console. Alternatively, after you create your predictive scaling policy, inspect the `LoadForecast` and `CapacityForecast` objects returned by a call to the [GetPredictiveScalingForecast](#) API.
- We strongly recommend that you configure predictive scaling in forecast only mode so that you can evaluate the forecast before predictive scaling starts actively scaling capacity.

Prerequisites

To add custom metrics to your predictive scaling policy, you must have `cloudwatch:GetMetricData` permissions.

To specify your own metrics instead of the metrics that AWS provides, you must first publish your metrics to CloudWatch. For more information, see [Publishing custom metrics](#) in the *Amazon CloudWatch User Guide*.

If you publish your own metrics, make sure to publish the data points at a minimum frequency of five minutes. Amazon EC2 Auto Scaling retrieves the data points from CloudWatch based on the length of the period that it needs. For example, the load metric specification uses hourly metrics to measure the load on your application. CloudWatch uses your published metric data to provide a single data value for any one-hour period by aggregating all data points with timestamps that fall within each one-hour period.

Constructing the JSON for custom metrics

The following section contains examples for how to configure predictive scaling to query data from CloudWatch. There are two different methods to configure this option, and the method that you choose affects which format you use to construct the JSON for your predictive scaling policy. When you use metric math, the format of the JSON varies further based on the metric math being performed.

1. To create a policy that gets data directly from other CloudWatch metrics provided by AWS or metrics that you publish to CloudWatch, see [Example predictive scaling policy with custom load and scaling metrics \(AWS CLI\) \(p. 184\)](#).
2. To create a policy that can query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics, see [Use metric math expressions \(p. 185\)](#).

Example predictive scaling policy with custom load and scaling metrics (AWS CLI)

To create a predictive scaling policy with custom load and scaling metrics with the AWS CLI, store the arguments for `--predictive-scaling-configuration` in a JSON file named `config.json`.

You start adding custom metrics by replacing the replaceable values in the following example with those of your metrics and your target utilization.

```
{  
  "MetricSpecifications": [  
    {  
      "TargetValue": 50,  
      "CustomizedScalingMetricSpecification": {  
        "MetricDataQueries": [  
          {  
            "MetricStat": {  
              "Metric": {  
                "MetricName": "MyUtilizationMetric",  
                "Namespace": "MyNameSpace",  
                "Dimensions": [  
                  {  
                    "Name": "MyOptionalMetricDimensionName",  
                    "Value": "MyOptionalMetricDimensionValue"  
                  }  
                ]  
              },  
              "Stat": "Average"  
            }  
          ]  
        }  
      }  
    ]  
  ]  
}
```

```

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

```

For more information, see [MetricDataQuery](#) in the *Amazon EC2 Auto Scaling API Reference*.

Note

Following are some additional resources that can help you find metric names, namespaces, dimensions, and statistics for CloudWatch metrics:

- For information about the available metrics for AWS services, see [AWS services that publish CloudWatch metrics](#) in the *Amazon CloudWatch User Guide*.
- To get the exact metric name, namespace, and dimensions (if applicable) for a CloudWatch metric with the AWS CLI, see [list-metrics](#).

To create this policy, run the `put-scaling-policy` command using the JSON file as input, as demonstrated in the following example.

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

If successful, this command returns the policy's Amazon Resource Name (ARN).

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

Use metric math expressions

The following section provides information and examples of predictive scaling policies that show how you can use metric math in your policy.

Contents

- [Understand metric math \(p. 186\)](#)

- [Example predictive scaling policy that combines metrics using metric math \(AWS CLI\) \(p. 186\)](#)
- [Example predictive scaling policy to use in a blue/green deployment scenario \(AWS CLI\) \(p. 188\)](#)

Understand metric math

If all you want to do is aggregate existing metric data, CloudWatch metric math saves you the effort and cost of publishing another metric to CloudWatch. You can use any metric that AWS provides, and you can also use metrics that you define as part of your applications. For example, you might want to calculate the Amazon SQS queue backlog per instance. You can do this by taking the approximate number of messages available for retrieval from the queue and dividing that number by the Auto Scaling group's running capacity.

For more information, see [Using metric math](#) in the *Amazon CloudWatch User Guide*.

If you choose to use a metric math expression in your predictive scaling policy, consider the following points:

- Metric math operations use the data points of the unique combination of metric name, namespace, and dimension keys/value pairs of metrics.
- You can use any arithmetic operator (+ - * / ^), statistical function (such as AVG or SUM), or other function that CloudWatch supports.
- You can use both metrics and the results of other math expressions in the formulas of the math expression.
- Your metric math expressions can be made up of different aggregations. However, it's a best practice for the final aggregation result to use Average for the scaling metric and Sum for the load metric.
- Any expressions used in a metric specification must eventually return a single time series.

To use metric math, do the following:

- Choose one or more CloudWatch metrics. Then, create the expression. For more information, see [Using metric math](#) in the *Amazon CloudWatch User Guide*.
- Verify that the metric math expression is valid by using the CloudWatch console or the CloudWatch [GetMetricData](#) API.

Example predictive scaling policy that combines metrics using metric math (AWS CLI)

Sometimes, instead of specifying the metric directly, you might need to first process its data in some way. For example, you might have an application that pulls work from an Amazon SQS queue, and you might want to use the number of items in the queue as criteria for predictive scaling. The number of messages in the queue does not solely define the number of instances that you need. Therefore, more work is needed to create a metric that can be used to calculate the backlog per instance. For more information, see [Scaling based on Amazon SQS \(p. 157\)](#).

The following is an example predictive scaling policy for this scenario. It specifies scaling and load metrics that are based on the Amazon SQS `ApproximateNumberOfMessagesVisible` metric, which is the number of messages available for retrieval from the queue. It also uses the Amazon EC2 Auto Scaling `GroupInServiceInstances` metric and a math expression to calculate the backlog per instance for the scaling metric.

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
--auto-scaling-group-name my-asg --policy-type PredictiveScaling \
--predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
```

```
{
  "TargetValue": 100,
  "CustomizedScalingMetricSpecification": {
    "MetricDataQueries": [
      {
        "Label": "Get the queue size (the number of messages waiting to be processed)",
        "Id": "queue_size",
        "MetricStat": {
          "Metric": {
            "MetricName": "ApproximateNumberOfMessagesVisible",
            "Namespace": "AWS/SQS",
            "Dimensions": [
              {
                "Name": "QueueName",
                "Value": "my-queue"
              }
            ],
            "Stat": "Sum"
          },
          "ReturnData": false
        },
        {
          "Label": "Get the group size (the number of running instances)",
          "Id": "running_capacity",
          "MetricStat": {
            "Metric": {
              "MetricName": "GroupInServiceInstances",
              "Namespace": "AWS/AutoScaling",
              "Dimensions": [
                {
                  "Name": "AutoScalingGroupName",
                  "Value": "my-asg"
                }
              ],
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
                "Stat": "Sum"
              },
              "ReturnData": true
            }
          }
        ]
      }
    ]
  }
}
```

]} }] }

The example returns the policy's ARN.

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

Example predictive scaling policy to use in a blue/green deployment scenario (AWS CLI)

A search expression provides an advanced option in which you can query for a metric from multiple Auto Scaling groups and perform math expressions on them. This is especially useful for blue/green deployments.

Note

A blue/green deployment is a deployment method in which you create two separate but identical Auto Scaling groups. Only one of the groups receives production traffic. User traffic is initially directed to the earlier ("blue") Auto Scaling group, while a new group ("green") is used for testing and evaluation of a new version of an application or service. User traffic is shifted to the green Auto Scaling group after a new deployment is tested and accepted. You can then delete the blue group after the deployment is successful.

When new Auto Scaling groups get created as part of a blue/green deployment, the metric history of each group can be automatically included in the predictive scaling policy without you having to change its metric specifications. For more information, see [Using EC2 Auto Scaling predictive scaling policies with Blue/Green deployments](#) on the AWS Compute Blog.

The following example policy shows how this can be done. In this example, the policy uses the `CPUUtilization` metric emitted by Amazon EC2. It uses the `Amazon EC2 Auto Scaling GroupInServiceInstances` metric and a math expression to calculate the value of the scaling metric per instance. It also specifies a capacity metric specification to get the `GroupInServiceInstances` metric.

The search expression finds the `CPUUtilization` of instances in multiple Auto Scaling groups based on the specified search criteria. If you later create a new Auto Scaling group that matches the same search criteria, the `CPUUtilization` of the instances in the new Auto Scaling group is automatically included.

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

```

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

```

The example returns the policy's ARN.

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

Considerations and troubleshooting

If an issue occurs while using custom metrics, we recommend that you do the following:

- If an error message is provided, read the message and resolve the issue it reports, if possible.
- If an issue occurs when you are trying to use a search expression in a blue/green deployment scenario, first make sure that you understand how to create a search expression that looks for a partial match instead of an exact match. Also, check that your query finds only the Auto Scaling groups that are running the specific application. For more information about the search expression syntax, see [CloudWatch search expression syntax](#) in the *Amazon CloudWatch User Guide*.
- If you did not validate an expression in advance, the [put-scaling-policy](#) command validates it when you create your scaling policy. However, there is a possibility that this command might fail to identify the exact cause of the detected errors. To fix the issues, troubleshoot the errors that you receive in a response from a request to the [get-metric-data](#) command. You can also troubleshoot the expression from the CloudWatch console.
- When you view your **Load** and **Capacity** graphs in the console, the **Capacity** graph might not show any data. To ensure that the graphs have complete data, make sure that you consistently enable group

metrics for your Auto Scaling groups. For more information, see [Enable Auto Scaling group metrics \(console\) \(p. 264\)](#).

- The capacity metric specification is only useful for blue/green deployments when you have applications that run in different Auto Scaling groups over their lifetime. This custom metric lets you provide the total capacity of multiple Auto Scaling groups. Predictive scaling uses this to show historical data in the **Capacity** graphs in the console.
- You must specify `false` for `ReturnData` if `MetricDataQueries` specifies the `SEARCH()` function on its own without a math function like `SUM()`. This is because search expressions might return multiple time series, and a metric specification based on an expression can return only one time series.
- All metrics involved in a search expression should be of the same resolution.

Limitations

- You can query data points of up to 10 metrics in one metric specification.
- For the purposes of this limit, one expression counts as one metric.

Scheduled scaling for Amazon EC2 Auto Scaling

Scheduled scaling helps you to set up your own scaling schedule according to predictable load changes. For example, let's say that every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can configure a schedule for Amazon EC2 Auto Scaling to increase capacity on Wednesday and decrease capacity on Friday.

To use scheduled scaling, you create *scheduled actions*. Scheduled actions are performed automatically as a function of date and time. When you create a scheduled action, you specify when the scaling activity should occur and the new desired, minimum, and maximum sizes for the scaling action. You can create scheduled actions that scale one time only or that scale on a recurring schedule.

Contents

- [Considerations \(p. 190\)](#)
- [Recurring schedules \(p. 191\)](#)
- [Create and manage scheduled actions \(console\) \(p. 191\)](#)
- [Create and manage scheduled actions \(AWS CLI\) \(p. 193\)](#)
- [Limitations \(p. 195\)](#)

Considerations

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

- A scheduled action sets the desired, minimum, and maximum sizes to what is specified by the scheduled action at the date and time specified. The request can optionally include only one of these sizes. For example, you can create a scheduled action with only the desired capacity specified. In some cases, however, you must include the minimum and maximum sizes to ensure that the new desired capacity that you specified in the action is not outside of these limits.
- By default, the recurring schedules that you set are in Coordinated Universal Time (UTC). You can change the time zone to correspond to your local time zone or a time zone for another part of your network. When you specify a time zone that observes Daylight Saving Time (DST), the action automatically adjusts for DST.
- You can temporarily turn off scheduled scaling for an Auto Scaling group by suspending the `ScheduledActions` process. This helps you prevent scheduled actions from being active without

having to delete them. You can then resume scheduled scaling when you want to use it again. For more information, see [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#).

- The order of execution for scheduled actions is guaranteed within the same group, but not for scheduled actions across groups.
- A scheduled action generally executes within seconds. However, the action might be delayed for up to two minutes from the scheduled start time. Because scheduled actions within an Auto Scaling group are executed in the order that they are specified, actions with scheduled start times close to each other can take longer to execute.

Recurring schedules

You can create scheduled actions that scale your Auto Scaling group on a recurring schedule.

To create a recurring schedule using the AWS CLI or an SDK, specify a cron expression and a time zone to describe when that scheduled action is to recur. You can optionally specify a date and time for the start time, the end time, or both.

To create a recurring schedule using the AWS Management Console, specify the recurrence pattern, time zone, start time, and optional end time of your scheduled action. All of the recurrence pattern options are based on cron expressions. Alternatively, you can write your own custom cron expression.

The supported cron expression format consists of five fields separated by white spaces: [Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]. For example, the cron expression 30 6 * * 2 configures a scheduled action that recurs every Tuesday at 6:30 AM. The asterisk is used as a wildcard to match all values for a field. For other examples of cron expressions, see <https://crontab.guru/examples.html>. For information about writing your own cron expressions in this format, see [Crontab](#).

Choose your start and end times carefully. Keep the following in mind:

- If you specify a start time, Amazon EC2 Auto Scaling performs the action at this time, and then performs the action based on the specified recurrence.
- If you specify an end time, the action stops repeating after this time. A scheduled action does not persist in your account once it has reached its end time.
- The start time and end time must be set in UTC when you use the AWS CLI or an SDK.

Create and manage scheduled actions (console)

Use the procedures in this section to create and manage scheduled actions using the AWS Management Console.

If you create a scheduled action using the console and specify a time zone that observes Daylight Saving Time (DST), both the recurring schedule and the start and end times automatically adjust for DST.

Create a scheduled action

Complete the following procedure to create a scheduled action to scale your Auto Scaling group.

To create a scheduled action for an Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Automatic scaling** tab, in **Scheduled actions**, choose **Create scheduled action**.
4. Enter a **Name** for the scheduled action.
5. For **Desired capacity, Min, Max**, choose the new desired size of the group and the new minimum and maximum capacity.
6. For **Recurrence**, choose one of the available options.
 - If you want to scale on a recurring schedule, choose how often Amazon EC2 Auto Scaling should run the scheduled action.
 - If you choose an option that begins with **Every**, the cron expression is created for you.
 - If you choose **Cron**, enter a cron expression that specifies when to perform the action.
 - If you want to scale only once, choose **Once**.
7. For **Time zone**, choose a time zone. The default is Etc/UTC.

Note
All of the time zones listed are from the IANA Time Zone database. For more information, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.
8. Define a date and time for **Specific start time**.
 - If you chose a recurring schedule, the start time defines when the first scheduled action in the recurring series runs.
 - If you chose **Once** as the recurrence, the start time defines the date and time for the schedule action to run.
9. (Optional) For recurring schedules, you can specify an end time by choosing **Set End Time** and then choosing a date and time for **End by**.
10. Choose **Create**. The console displays the scheduled actions for the Auto Scaling group.

Verify the time, date, and time zone

To verify whether your time, date, and time zone are configured correctly, check the **Start time**, **End time**, and **Time zone** values in the **Scheduled actions** table on the **Automatic scaling** tab for your Auto Scaling group.

Amazon EC2 Auto Scaling shows the values for **Start time** and **End time** in your local time with the UTC offset in effect at the specified date and time. The UTC offset is the difference, in hours and minutes, from local time to UTC. The value for **Time zone** shows your requested time zone, for example, America/New_York.

Location-based time zones such as America/New_York automatically adjust for Daylight Savings Time (DST). However, a UTC-based time zone such as Etc/UTC is an absolute time and will not adjust for DST.

For example, you have a recurring schedule whose time zone is America/New_York. The first scaling action happens in the America/New_York time zone before DST starts. The next scaling action happens in the America/New_York time zone after DST starts. The first action starts at 8:00 AM UTC-5 in local time, while the second time starts at 8:00 AM UTC-4 in local time.

Update a scheduled action

After creating a scheduled action, you can update any of its settings except the name.

To update a scheduled action

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action.
4. Choose **Actions, Edit**.
5. Make the needed changes and choose **Save changes**.

Delete a scheduled action

When you no longer need a scheduled action, you can delete it.

To delete a scheduled action

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select your Auto Scaling group.
3. On the **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action.
4. Choose **Actions, Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Create and manage scheduled actions (AWS CLI)

You can create and update scheduled actions that scale one time only or that scale on a recurring schedule using the [put-scheduled-update-group-action](#) command.

Create a scheduled action that occurs only once

To automatically scale your Auto Scaling group one time only, at a specified date and time, use the `--start-time "YYYY-MM-DDThh:mm:ssZ"` option.

Example: To scale out one time only

To increase the number of running instances in your Auto Scaling group at a specific time, use the following command.

At the date and time specified for `--start-time` (8:00 AM UTC on March 31, 2021), if the group currently has fewer than 3 instances, it scales out to 3 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
--auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

Example: To scale in one time only

To decrease the number of running instances in your Auto Scaling group at a specific time, use the following command.

At the date and time specified for `--start-time` (4:00 PM UTC on March 31, 2021), if the group currently has more than 1 instance, it scales in to 1 instance.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
--auto-scaling-group-name my-asg --start-time "2021-03-31T16:00:00Z" --desired-capacity 1
```

Create a scheduled action that runs on a recurring schedule

To schedule scaling on a recurring schedule, use the `--recurrence "cron expression"` option.

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

On the specified schedule (every day at 9:00 AM UTC), if the group currently has fewer than 3 instances, it scales out to 3 instances. If the group currently has more than 3 instances, it scales in to 3 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
--auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

Create a recurring scheduled action that specifies a time zone

Scheduled actions are set to the UTC time zone by default. To specify a different time zone, include the `--time-zone` option and specify the canonical name for the IANA time zone (`America/New_York`, for example). For more information, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

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

On the specified schedule (every Monday through Friday at 6:00 PM local time), if the group currently has fewer than 2 instances, it scales out to 2 instances. If the group currently has more than 2 instances, it scales in to 2 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
--auto-scaling-group-name my-asg --recurrence "0 18 * * 1-5" --time-zone "America/
New_York" \
--desired-capacity 2
```

Describe scheduled actions

To describe the scheduled actions for an Auto Scaling group, use the following [describe-scheduled-actions](#) command.

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

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

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

Delete a scheduled action

To delete a scheduled action, use the following [delete-scheduled-action](#) command.

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \  
--scheduled-action-name my-recurring-action
```

Limitations

- The names of scheduled actions must be unique per Auto Scaling group.
- A scheduled action must have a unique time value. If you attempt to schedule an activity at a time when another scaling activity is already scheduled, the call is rejected and returns an error indicating that a scheduled action with this scheduled start time already exists.
- You can create a maximum of 125 scheduled actions per Auto Scaling group.

Amazon EC2 Auto Scaling lifecycle hooks

Amazon EC2 Auto Scaling offers the ability to add lifecycle hooks to your Auto Scaling groups. These hooks let you create solutions that are aware of events in the Auto Scaling instance lifecycle, and then perform a custom action on instances when the corresponding lifecycle event occurs. A lifecycle hook provides a specified amount of time (one hour by default) to wait for the action to complete before the instance transitions to the next state.

As an example of using lifecycle hooks with Auto Scaling instances:

- When a scale-out event occurs, your newly launched instance completes its startup sequence and transitions to a wait state. While the instance is in a wait state, it runs a script to download and install the needed software packages for your application, making sure that your instance is fully ready before it starts receiving traffic. When the script is finished installing software, it sends the **complete-lifecycle-action** command to continue.
- When a scale-in event occurs, a lifecycle hook pauses the instance before it is terminated and sends you a notification using Amazon EventBridge. While the instance is in the wait state, you can invoke an AWS Lambda function or connect to the instance to download logs or other data before the instance is fully terminated.

A popular use of lifecycle hooks is to control when instances are registered with Elastic Load Balancing. By adding a launch lifecycle hook to your Auto Scaling group, you can ensure that your bootstrap scripts have completed successfully and the applications on the instances are ready to accept traffic before they are registered to the load balancer at the end of the lifecycle hook.

For an introduction video, see [AWS re:Invent 2018: Capacity Management Made Easy with Amazon EC2 Auto Scaling](#) on [YouTube](#).

Contents

- [Considerations and limitations for lifecycle hooks \(p. 196\)](#)
- [Lifecycle hook availability \(p. 197\)](#)
- [Examples \(p. 197\)](#)
- [How lifecycle hooks work \(p. 197\)](#)
- [Prepare to add a lifecycle hook to your Auto Scaling group \(p. 199\)](#)
- [Add lifecycle hooks \(p. 205\)](#)
- [Complete a lifecycle action \(p. 207\)](#)

- [Tutorial: Configure user data to retrieve the target lifecycle state through instance metadata \(p. 208\)](#)
- [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#)

Considerations and limitations for lifecycle hooks

When using lifecycle hooks, keep in mind the following considerations and limitations:

- Amazon EC2 Auto Scaling provides its own lifecycle to help with the management of Auto Scaling groups. This lifecycle differs from that of other EC2 instances. For more information, see [Amazon EC2 Auto Scaling instance lifecycle \(p. 7\)](#).
- Instances in a warm pool also have their own lifecycle, as described in [Lifecycle state transitions for instances in a warm pool \(p. 224\)](#).
- You can use lifecycle hooks with Spot Instances, but a lifecycle hook does not prevent an instance from terminating in the event that capacity is no longer available, which can happen at any time with a two-minute interruption notice. For more information, see [Spot Instance interruptions](#) in the *Amazon EC2 User Guide for Linux Instances*. However, you can enable Capacity Rebalancing to proactively replace Spot Instances that have received a rebalance recommendation from the Amazon EC2 Spot service, a signal that is sent when a Spot Instance is at elevated risk of interruption. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).
- When scaling out, Amazon EC2 Auto Scaling doesn't count a new instance towards the aggregated CloudWatch instance metrics of the Auto Scaling group (such as CPUUtilization, NetworkIn, NetworkOut, and so on) until after the launch lifecycle hook finishes. When default instance warmup is not enabled, or enabled but set to 0, Auto Scaling instances start contributing usage data to the aggregated instance metrics as soon as they reach the InService state. For more information, see [Set the default instance warmup for an Auto Scaling group \(p. 150\)](#).

On scale in, the aggregated instance metrics might not instantly reflect the removal of a terminating instance. The terminating instance stops counting toward the group's aggregated instance metrics shortly after the Amazon EC2 Auto Scaling termination workflow begins.

- When an Auto Scaling group launches or terminates instances, scaling activities initiated by simple scaling policies are paused. If lifecycle hooks are invoked, scaling activities due to simple scaling policies are paused until the lifecycle actions have completed and the cooldown period has expired. Setting a long interval for the cooldown period means that it will take longer for scaling to resume. For more information, see [Scaling cooldowns for Amazon EC2 Auto Scaling \(p. 153\)](#).
- Instances can remain in a wait state for a finite period of time. The default timeout for a lifecycle hook is one hour (heartbeat timeout). There is also a global timeout that specifies the maximum amount of time that you can keep an instance in a wait state. The global timeout is 48 hours or 100 times the heartbeat timeout, whichever is smaller.

When creating lifecycle hooks, keep in mind the following points:

- You can configure a launch lifecycle hook to abandon the launch if an unexpected failure occurs, in which case Amazon EC2 Auto Scaling automatically terminates and replaces the instance.
- Amazon EC2 Auto Scaling limits the rate at which it allows instances to launch if the lifecycle hooks are failing consistently, so make sure to test and fix any permanent errors in your lifecycle actions.
- Creating and updating lifecycle hooks using the AWS CLI, AWS CloudFormation, or an SDK provides options not available when creating a lifecycle hook from the AWS Management Console. For example, the field to specify the ARN of an SNS topic or SQS queue doesn't appear in the console, because Amazon EC2 Auto Scaling already sends events to Amazon EventBridge. These events can be filtered and redirected to AWS services such as Lambda, Amazon SNS, and Amazon SQS as needed.
- You can add multiple lifecycle hooks to an Auto Scaling group while you are creating it, by calling the [CreateAutoScalingGroup](#) API using the AWS CLI, AWS CloudFormation, or an SDK. However, each

hook must have the same notification target and IAM role, if specified. To create lifecycle hooks with different notification targets and different roles, create the lifecycle hooks one at a time in separate calls to the [PutLifecycleHook](#) API.

Lifecycle hook availability

The following table lists the lifecycle hooks available for various scenarios.

Event	Instance launch or termination ¹	Maximum Instance Lifetime: Replacement instances	Instance Refresh: Replacement instances	Capacity Rebalancing: Replacement instances	Warm Pools: Instances entering and leaving the warm pool
Instance launching	✓	✓	✓	✓	✓
Instance terminating	✓	✓	✓	✓	✓

¹ Applies to instances launched or terminated when the group is created or deleted, when the group scales automatically, or when you manually adjust your group's desired capacity. Does not apply when you attach or detach instances, move instances in and out of standby mode, or delete the group with the force delete option.

Examples

We provide a few JSON and YAML template snippets that you can use to understand how to declare lifecycle hooks in your AWS CloudFormation stack templates. For more information, see the [AWS::AutoScaling::LifecycleHook](#) reference in the *AWS CloudFormation User Guide*.

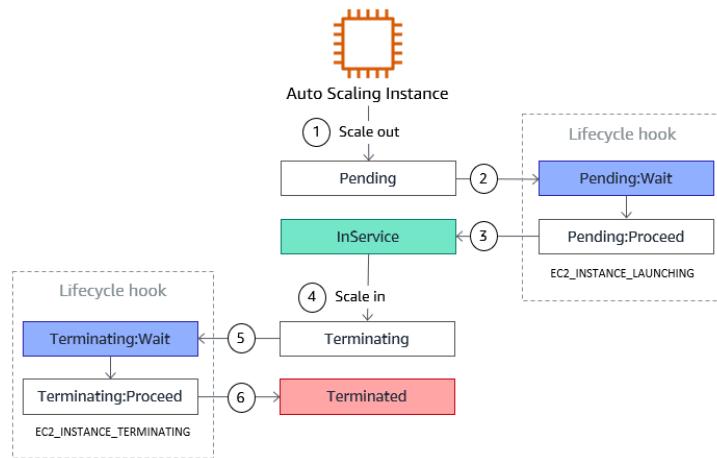
You can also visit our [GitHub repository](#) to download example templates and user data scripts for lifecycle hooks.

For other examples of the use of lifecycle hooks, see the following blog posts: [Building a Backup System for Scaled Instances using Lambda and Amazon EC2 Run Command](#) and [Run code before terminating an EC2 Auto Scaling instance](#).

How lifecycle hooks work

An Amazon EC2 instance transitions through different states from the time it launches until it is terminated. You can create lifecycle hooks to act when an instance transitions into a wait state.

The following illustration shows the transitions between Auto Scaling instance states.



As shown in the preceding diagram:

1. The Auto Scaling group responds to a scale-out event and begins launching an instance.
2. The lifecycle hook puts the instance into a wait state (Pending:Wait) and then performs a custom action.

The instance remains in a wait state until you either complete the lifecycle action, or the timeout period ends. By default, the instance remains in a wait state for one hour, and then the Auto Scaling group continues the launch process (Pending:Proceed). If you need more time, you can restart the timeout period by recording a heartbeat. If you complete the lifecycle action when the custom action has completed and the timeout period hasn't expired yet, the period ends and the Auto Scaling group continues the launch process.

3. The instance enters the InService state and the health check grace period starts. However, before the instance reaches the InService state, if the Auto Scaling group is associated with an Elastic Load Balancing load balancer, the instance is registered with the load balancer, and the load balancer starts checking its health. After the health check grace period ends, Amazon EC2 Auto Scaling begins checking the health state of the instance.
4. The Auto Scaling group responds to a scale-in event and begins terminating an instance. If the Auto Scaling group is being used with Elastic Load Balancing, the terminating instance is first deregistered from the load balancer. If connection draining is enabled for the load balancer, the instance stops accepting new connections and waits for existing connections to drain before completing the deregistration process.
5. The lifecycle hook puts the instance into a wait state (Terminating:Wait) and then performs a custom action.

The instance remains in a wait state either until you complete the lifecycle action, or until the timeout period ends (one hour by default). After you complete the lifecycle hook or the timeout period expires, the instance transitions to the next state (Terminating:Proceed).

6. The instance is terminated.

Important

Instances in a warm pool also have their own lifecycle with corresponding wait states, as described in [Lifecycle state transitions for instances in a warm pool \(p. 224\)](#).

Prepare to add a lifecycle hook to your Auto Scaling group

Before you add a lifecycle hook to your Auto Scaling group, be sure that your user data script or notification target is set up correctly.

- To use a user data script to perform custom actions on your instances as they are launching, you do not need to configure a notification target. However, you must have already created the launch template or launch configuration that specifies your user data script and associated it with your Auto Scaling group. For more information about user data scripts, see [Run commands on your Linux instance at launch](#) in the *Amazon EC2 User Guide for Linux Instances*.
- To signal Amazon EC2 Auto Scaling when the lifecycle action is complete, you must add the [CompleteLifecycleAction](#) API call to the script, and you must manually create an IAM role with a policy that allows Auto Scaling instances to call this API. Your launch template or launch configuration must specify this role using an IAM instance profile that gets attached to your Amazon EC2 instances at launch. For more information, see [Complete a lifecycle action \(p. 207\)](#) and [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).
- To use a service such as Lambda to perform a custom action, you must have already created an EventBridge rule and specified a Lambda function as its target. For more information, see [Configure a notification target for lifecycle notifications \(p. 199\)](#).
- To allow Lambda to signal Amazon EC2 Auto Scaling when the lifecycle action is complete, you must add the [CompleteLifecycleAction](#) API call to the function code. You must also have attached an IAM policy to the function's execution role that gives Lambda permission to complete lifecycle actions. For more information, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#).
- To use a service such as a Amazon SNS or Amazon SQS to perform a custom action, you must have already created the SNS topic or SQS queue and have ready its Amazon Resource Name (ARN). You must also have already created the IAM role that gives Amazon EC2 Auto Scaling access to your SNS topic or SQS target and have ready its ARN. For more information, see [Configure a notification target for lifecycle notifications \(p. 199\)](#).

Note

By default, when you add a lifecycle hook in the console, Amazon EC2 Auto Scaling sends lifecycle event notifications to Amazon EventBridge. Using EventBridge or a user data script is a recommended best practice. To create a lifecycle hook that sends notifications directly to Amazon SNS or Amazon SQS, use the AWS CLI, AWS CloudFormation, or an SDK to add the lifecycle hook.

Configure a notification target for lifecycle notifications

You can add lifecycle hooks to an Auto Scaling group to perform custom actions when an instance enters a wait state. You can choose a target service to perform these actions depending on your preferred development approach.

The first approach uses Amazon EventBridge to invoke a Lambda function that performs the action you want. The second approach involves creating an Amazon Simple Notification Service (Amazon SNS) topic to which notifications are published. Clients can subscribe to the SNS topic and receive published messages using a supported protocol. The last approach involves using Amazon Simple Queue Service (Amazon SQS), a messaging system used by distributed applications to exchange messages through a polling model.

As a best practice, we recommend that you use EventBridge. The notifications sent to Amazon SNS and Amazon SQS contain the same information as the notifications that Amazon EC2 Auto Scaling sends to EventBridge. Before EventBridge, the standard practice was to send a notification to SNS or SQS and integrate another service with SNS or SQS to perform programmatic actions. Today, EventBridge gives

you more options for which services you can target and makes it easier to handle events using serverless architecture.

The following procedures cover how to set up your notification target.

Remember, if you have a user data script in your launch template or launch configuration that configures your instances when they launch, you do not need to receive notifications to perform custom actions on your instances.

Contents

- [Route notifications to Lambda using EventBridge \(p. 200\)](#)
- [Receive notifications using Amazon SNS \(p. 202\)](#)
- [Receive notifications using Amazon SQS \(p. 202\)](#)
- [Notification message example for Amazon SNS and Amazon SQS \(p. 203\)](#)

Important

The EventBridge rule, Lambda function, Amazon SNS topic, and Amazon SQS queue that you use with lifecycle hooks must always be in the same Region where you created your Auto Scaling group.

Route notifications to Lambda using EventBridge

You can configure an EventBridge rule to invoke a Lambda function when an instance enters a wait state. Amazon EC2 Auto Scaling emits a lifecycle event notification to EventBridge about the instance that is launching or terminating and a token that you can use to control the lifecycle action. For examples of these events, see [Amazon EC2 Auto Scaling event reference \(p. 304\)](#).

Note

When you use the AWS Management Console to create an event rule, the console automatically adds the IAM permissions necessary to grant EventBridge permission to call your Lambda function. If you are creating an event rule using the AWS CLI, you need to grant this permission explicitly.

For information about how to create event rules in the EventBridge console, see [Creating Amazon EventBridge rules that react to events](#) in the *Amazon EventBridge User Guide*.

– or –

For an introductory tutorial that is directed towards console users, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#). This tutorial shows you how to create a simple Lambda function that listens for launch events and writes them out to a CloudWatch Logs log.

To create an EventBridge rule that invokes a Lambda function

1. Create a Lambda function by using the [Lambda console](#) and note its Amazon Resource Name (ARN). For example, `arn:aws:lambda:region:123456789012:function:my-function`. You need the ARN to create an EventBridge target. For more information, see [Getting started with Lambda](#) in the *AWS Lambda Developer Guide*.
2. To create a rule that matches events for instance launch, use the following [put-rule](#) command.

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state ENABLED
```

The following example shows the `pattern.json` for an instance launch lifecycle action. Replace the text in *italics* with the name of your Auto Scaling group.

```
{
```

```
"source": [ "aws.autoscaling" ],
"detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
"detail": {
    "AutoScalingGroupName": [ "my-asg" ]
}
```

If the command runs successfully, EventBridge responds with the ARN of the rule. Note this ARN. You'll need to enter it in step 4.

To create a rule that matches for other events, modify the event pattern. For more information, see [Use EventBridge to handle Auto Scaling events \(p. 304\)](#).

3. To specify the Lambda function to use as a target for the rule, use the following [put-targets](#) command.

```
aws events put-targets --rule my-rule --targets
  Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

In the preceding command, *my-rule* is the name that you specified for the rule in step 2, and the value for the *Arn* parameter is the ARN of the function that you created in step 1.

4. To add permissions that allow the rule to invoke your Lambda function, use the following Lambda [add-permission](#) command. This command trusts the EventBridge service principal (events.amazonaws.com) and scopes permissions to the specified rule.

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
  arn:aws:events:region:123456789012:rule/my-rule
```

In the preceding command:

- *my-function* is the name of the Lambda function that you want the rule to use as a target.
- *my-unique-id* is a unique identifier that you define to describe the statement in the Lambda function policy.
- *source-arn* is the ARN of the EventBridge rule.

If the command runs successfully, you receive output similar to the following.

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":{
      \"ArnLike\":{
        \"AWS:SourceArn\":
          \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"}
```

The *Statement* value is a JSON string version of the statement that was added to the Lambda function policy.

5. After you have followed these instructions, continue on to [Add lifecycle hooks \(p. 205\)](#) as a next step.

Receive notifications using Amazon SNS

You can use Amazon SNS to set up a notification target (an SNS topic) to receive notifications when a lifecycle action occurs. Amazon SNS then sends the notifications to the subscribed recipients. Until the subscription is confirmed, no notifications published to the topic are sent to the recipients.

To set up notifications using Amazon SNS

1. Create an Amazon SNS topic by using either the [Amazon SNS console](#) or the following [create-topic](#) command. Ensure that the topic is in the same Region as the Auto Scaling group that you're using. For more information, see [Getting started with Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

```
aws sns create-topic --name my-sns-topic
```

2. Note the topic Amazon Resource Name (ARN), for example, `arn:aws:sns:region:123456789012:my-sns-topic`. You need it to create the lifecycle hook.
3. Create an IAM service role to give Amazon EC2 Auto Scaling access to your Amazon SNS notification target.

To give Amazon EC2 Auto Scaling access to your SNS topic

- a. Open the IAM console at <https://console.aws.amazon.com/iam/>.
 - b. In the navigation pane on the left, choose **Roles**.
 - c. Choose **Create role**.
 - d. For **Select trusted entity**, choose **AWS service**.
 - e. For your use case, under **Use cases for other AWS services**, choose **EC2 Auto Scaling** and then **EC2 Auto Scaling Notification Access**.
 - f. Choose **Next** twice to go to the **Name, review, and create** page.
 - g. For **Role name**, enter a name for the role (for example, **my-notification-role**) and choose **Create role**.
 - h. On the **Roles** page, choose the role that you just created to open the **Summary** page. Make a note of the role **ARN**. For example, `arn:aws:iam::123456789012:role/my-notification-role`. You need it to create the lifecycle hook.
4. After you have followed these instructions, continue on to [Add lifecycle hooks \(AWS CLI\) \(p. 206\)](#) as a next step.

Receive notifications using Amazon SQS

You can use Amazon SQS to set up a notification target to receive messages when a lifecycle action occurs. A queue consumer must then poll an SQS queue to act on these notifications.

Important

FIFO queues are not compatible with lifecycle hooks.

To set up notifications using Amazon SQS

1. Create an Amazon SQS queue by using the [Amazon SQS console](#). Ensure that the queue is in the same Region as the Auto Scaling group that you're using. For more information, see [Getting started with Amazon SQS](#) in the *Amazon Simple Queue Service Developer Guide*.
2. Note the queue ARN, for example, `arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`. You need it to create the lifecycle hook.
3. Create an IAM service role to give Amazon EC2 Auto Scaling access to your Amazon SQS notification target.

To give Amazon EC2 Auto Scaling access to your SQS queue

- a. Open the IAM console at <https://console.aws.amazon.com/iam/>.
 - b. In the navigation pane on the left, choose **Roles**.
 - c. Choose **Create role**.
 - d. For **Select trusted entity**, choose **AWS service**.
 - e. For your use case, under **Use cases for other AWS services**, choose **EC2 Auto Scaling** and then **EC2 Auto Scaling Notification Access**.
 - f. Choose **Next** twice to go to the **Name, review, and create** page.
 - g. For **Role name**, enter a name for the role (for example, **my-notification-role**) and choose **Create role**.
 - h. On the **Roles** page, choose the role that you just created to open the **Summary** page. Make a note of the role **ARN**. For example, `arn:aws:iam::123456789012:role/my-notification-role`. You need it to create the lifecycle hook.
4. After you have followed these instructions, continue on to [Add lifecycle hooks \(AWS CLI\) \(p. 206\)](#) as a next step.

Notification message example for Amazon SNS and Amazon SQS

While the instance is in a wait state, a message is published to the Amazon SNS or Amazon SQS notification target. The message includes the following information:

- **LifecycleActionToken** — The lifecycle action token.
- **AccountId** — The AWS account ID.
- **AutoScalingGroupName** — The name of the Auto Scaling group.
- **LifecycleHookName** — The name of the lifecycle hook.
- **EC2InstanceId** — The ID of the EC2 instance.
- **LifecycleTransition** — The lifecycle hook type.
- **NotificationMetadata** — The notification metadata.

The following is a notification message example.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

Test notification message example

When you first add a lifecycle hook, a test notification message is published to the notification target. The following is a test notification message example.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
```

```
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

Note

For examples of the events delivered from Amazon EC2 Auto Scaling to EventBridge, see [Amazon EC2 Auto Scaling event reference \(p. 304\)](#).

Retrieve the target lifecycle state through instance metadata

Important

Amazon EC2 Auto Scaling started generating the target lifecycle state on March 10, 2022. If your instance transitions to one of the target lifecycle states after that date, the target lifecycle state item is present in your instance metadata. Otherwise, it is not present, and you receive an HTTP 404 error.

Each Auto Scaling instance that you launch goes through several lifecycle states. If you want to fine tune your on-instance custom actions to act on specific lifecycle state transitions, you can do this by retrieving the target lifecycle state through instance metadata. For example, you might retrieve the target lifecycle state when creating lifecycle actions for instances in a warm pool so that hibernated or stopped instances can be attached to a cluster on restart.

The Auto Scaling instance lifecycle has two primary steady states—`InService` and `Terminated`—and two side steady states—`Detached` and `Standby`. If you use a warm pool, the lifecycle has four additional steady states—`Warmed:Hibernated`, `Warmed:Running`, `Warmed:Stopped`, and `Warmed:Terminated`.

When an instance prepares to transition to one of the preceding steady states, Amazon EC2 Auto Scaling updates the value of the instance metadata item `autoscaling/target-lifecycle-state`. To get the target lifecycle state from within the instance, you must use the instance metadata service to retrieve it from the instance metadata.

Note

Instance metadata is data about an Amazon EC2 instance that applications can use to query instance information. The *instance metadata service* is an on-instance component that local code uses to access instance metadata. Local code can include user data scripts or applications running on the instance.

Local code can access instance metadata from a running instance using one of two methods: Instance Metadata Service Version 1 (IMDSv1) or Instance Metadata Service Version 2 (IMDSv2). IMDSv2 uses session-oriented requests and mitigates several types of vulnerabilities that could be used to try to access the instance metadata. For details about these two methods, see [Use IMDSv2](#) in the *Amazon EC2 User Guide for Linux Instances*.

IMDSv2

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-
token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-
data/autoscaling/target-lifecycle-state
```

IMDSv1

```
curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

The following is example output.

InService

The target lifecycle state is the state that the instance is transitioning to. The current lifecycle state is the state that the instance is in. These can be the same after the lifecycle action is complete and the instance finishes its transition to the target lifecycle state. You cannot retrieve the current lifecycle state of the instance from the instance metadata.

For a tutorial that shows you how to create a lifecycle hook with a custom action in a user data script that uses the target lifecycle state, see [Tutorial: Configure user data to retrieve the target lifecycle state through instance metadata \(p. 208\)](#).

For more information about instance metadata categories that you can use to configure or manage instances, see [Instance metadata categories](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about retrieving instance metadata, see [Retrieve instance metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

Add lifecycle hooks

To put your Auto Scaling instances into a wait state and perform custom actions on them, you can add lifecycle hooks to your Auto Scaling group. Custom actions are performed as the instances launch or before they terminate. Instances remain in a wait state until you either complete the lifecycle action, or the timeout period ends.

After you create an Auto Scaling group from the AWS Management Console, you can add one or more lifecycle hooks to it, up to a total of 50 lifecycle hooks. You can also use the AWS CLI, AWS CloudFormation, or an SDK to add lifecycle hooks to an Auto Scaling group as you are creating it.

By default, when you add a lifecycle hook in the console, Amazon EC2 Auto Scaling sends lifecycle event notifications to Amazon EventBridge. Using EventBridge or a user data script is a recommended best practice. To create a lifecycle hook that sends notifications directly to Amazon SNS or Amazon SQS, you can use the [put-lifecycle-hook](#) command, as shown in the examples in this topic.

Contents

- [Add lifecycle hooks \(console\) \(p. 205\)](#)
- [Add lifecycle hooks \(AWS CLI\) \(p. 206\)](#)

Add lifecycle hooks (console)

Follow these steps to add a lifecycle hook to your Auto Scaling group. To create lifecycle hooks for scaling out (instances launching) and scaling in (instances terminating), you must create two separate hooks.

Before you begin, confirm that you have set up a custom action, as described in [Prepare to add a lifecycle hook to your Auto Scaling group \(p. 199\)](#).

To add a lifecycle hook

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. Select the check box next to your Auto Scaling group.
- A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. On the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.
 4. To define a lifecycle hook, do the following:
 - a. For **Lifecycle hook name**, specify a name for the lifecycle hook.

- b. For **Lifecycle transition**, choose **Instance launch** or **Instance terminate**.
- c. For **Heartbeat timeout**, specify the amount of time for instances to remain in a wait state when scaling out or scaling in before the hook times out. The range is from 30 to 7200 seconds.

Note

By default, the timeout is 3600 seconds (one hour). Setting a long timeout period provides more time for the custom action to complete. If you finish before the timeout period ends, you can send the [complete-lifecycle-action](#) command to allow the instance to proceed to the next state.

- d. For **Default result**, specify the action to take when the lifecycle hook timeout elapses or when an unexpected failure occurs. You can choose to either *abandon* (default) or *continue*.
 - If the instance is launching, *continue* indicates that your actions were successful, and that Amazon EC2 Auto Scaling can put the instance into service. Otherwise, *abandon* indicates that your custom actions were unsuccessful, and that Amazon EC2 Auto Scaling can terminate the instance.
 - If the instance is terminating, both *abandon* and *continue* allow the instance to terminate. However, *abandon* stops any remaining actions, such as other lifecycle hooks, and *continue* allows any other lifecycle hooks to complete.
- e. (Optional) For **Notification metadata**, specify additional information that you want to include when Amazon EC2 Auto Scaling sends a message to the notification target.

5. Choose **Create**.

Add lifecycle hooks (AWS CLI)

Create and update lifecycle hooks using the [put-lifecycle-hook](#) command.

To perform an action on scale out, use the following command.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook \
--auto-scaling-group-name my-asg \
--lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

To perform an action on scale in, use the following command instead.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \
--auto-scaling-group-name my-asg \
--lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

To receive notifications using Amazon SNS or Amazon SQS, add the `--notification-target-arn` and `--role-arn` options.

The following example creates a lifecycle hook that specifies an SNS topic named *my-sns-topic* as the notification target.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \
--auto-scaling-group-name my-asg \
--lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \
--notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \
--role-arn arn:aws:iam::123456789012:role/my-notification-role
```

The topic receives a test notification with the following key-value pair.

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

By default, the [put-lifecycle-hook](#) command creates a lifecycle hook with a heartbeat timeout of 3600 seconds (one hour).

To change the heartbeat timeout for an existing lifecycle hook, add the `--heartbeat-timeout` option, as shown in the following example.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \  
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```

If an instance is already in a wait state, you can prevent the lifecycle hook from timing out by recording a heartbeat, using the [record-lifecycle-action-heartbeat](#) CLI command. This extends the timeout period by the timeout value specified when you created the lifecycle hook. If you finish before the timeout period ends, you can send the [complete-lifecycle-action](#) CLI command to allow the instance to proceed to the next state. For more information and examples, see [Complete a lifecycle action \(p. 207\)](#).

Complete a lifecycle action

When an Auto Scaling group responds to a lifecycle event, it puts the instance in a wait state and sends an event notification. You can perform a custom action while the instance is in a wait state.

Contents

- [Complete a lifecycle action \(manual\) \(p. 207\)](#)
- [Complete a lifecycle action \(automatic\) \(p. 208\)](#)

Complete a lifecycle action (manual)

The following procedure is for the command line interface and is not supported in the console. Information that must be replaced, such as the instance ID or the name of an Auto Scaling group, are shown in italics.

To complete a lifecycle action (AWS CLI)

1. If you need more time to complete the custom action, use the [record-lifecycle-action-heartbeat](#) command to restart the timeout period and keep the instance in a wait state. For example, if the timeout period is one hour, and you call this command after 30 minutes, the instance remains in a wait state for an additional hour, or a total of 90 minutes.

You can specify the lifecycle action token that you received with the [notification \(p. 203\)](#), as shown in the following command.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  \  
  --auto-scaling-group-name my-asg --lifecycle-action-  
  token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Alternatively, you can specify the ID of the instance that you received with the [notification \(p. 203\)](#), as shown in the following command.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \  
  \  
  --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
```

2. If you finish the custom action before the timeout period ends, use the [complete-lifecycle-action](#) command so that the Auto Scaling group can continue launching or terminating the instance. You can specify the lifecycle action token, as shown in the following command.

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
  --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \  
  --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Alternatively, you can specify the ID of the instance, as shown in the following command.

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \  
  --instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \  
  --auto-scaling-group-name my-asg
```

Complete a lifecycle action (automatic)

If you have a user data script that configures your instances after they launch, you do not need to manually complete lifecycle actions. You can add the [complete-lifecycle-action](#) command to the script. The script can retrieve the instance ID from the instance metadata and signal Amazon EC2 Auto Scaling when the bootstrap scripts have completed successfully.

If you are not doing so already, update your script to retrieve the instance ID of the instance from the instance metadata. For more information, see [Retrieve instance metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you use Lambda, you can also set up a callback in your function's code to let the lifecycle of the instance proceed if the custom action is successful. For more information, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#).

Tutorial: Configure user data to retrieve the target lifecycle state through instance metadata

A common way to create custom actions for lifecycle hooks is to use notifications that Amazon EC2 Auto Scaling sends to other services, such as Amazon EventBridge. However, you can avoid having to create additional infrastructure by instead using a user data script to move the code that configures instances and completes the lifecycle action into the instances themselves.

The following tutorial shows you how to get started using a user data script and instance metadata. You create a basic Auto Scaling group configuration with a user data script that reads the [target lifecycle state \(p. 204\)](#) of the instances in your group and performs a callback action at a specific phase of an instance's lifecycle to continue the launch process.

Contents

- [Step 1: Create an IAM role with permissions to complete lifecycle actions \(p. 208\)](#)
- [Step 2: Create a launch template and include the IAM role and a user data script \(p. 209\)](#)
- [Step 3: Create an Auto Scaling group \(p. 211\)](#)
- [Step 4: Add a lifecycle hook \(p. 211\)](#)
- [Step 5: Test and verify the functionality \(p. 212\)](#)
- [Step 6: Clean up \(p. 212\)](#)

Step 1: Create an IAM role with permissions to complete lifecycle actions

When you use the AWS CLI or an AWS SDK to send a callback to complete lifecycle actions, you must use an IAM role with permissions to complete lifecycle actions.

To create the policy

1. Open the [Policies page](#) of the IAM console, and then choose **Create policy**.
2. Choose the **JSON** tab.
3. In the **Policy Document** box, copy and paste the following policy document into the box. Replace the *sample text* with your account number and the name of the Auto Scaling group that you want to create (**TestAutoScalingEvent-group**).

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "autoscaling:CompleteLifecycleAction"  
      ],  
      "Resource":  
        "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/  
        TestAutoScalingEvent-group"  
    }  
  ]  
}
```

4. Choose **Next:Tags**, and then **Next:Review**.
5. For **Name**, enter **TestAutoScalingEvent-policy**. Choose **Create policy**.

When you finish creating the policy, you can create a role that uses it.

To create the role

1. In the navigation pane on the left, choose **Roles**.
2. Choose **Create role**.
3. For **Select trusted entity**, choose **AWS service**.
4. For your use case, choose **EC2** and then choose **Next**.
5. Under **Add permissions**, choose the policy that you created (**TestAutoScalingEvent-policy**). Then, choose **Next**.
6. On the **Name, review, and create** page, for **Role name**, enter **TestAutoScalingEvent-role** and choose **Create role**.

Step 2: Create a launch template and include the IAM role and a user data script

Create a launch template to use with your Auto Scaling group. Include the IAM role you created and the provided sample user data script.

To create a launch template

1. Open the [Launch templates page](#) on the Amazon EC2 console.
2. Choose **Create launch template**.
3. For **Launch template name**, enter **TestAutoScalingEvent-template**.
4. Under **Auto Scaling guidance**, select the check box.
5. For **Application and OS Images (Amazon Machine Image)**, choose Amazon Linux 2 (HVM), SSD Volume Type, 64-bit (x86) from the **Quick Start** list.

6. For **Instance type**, choose a type of Amazon EC2 instance (for example, "t2.micro").
7. For **Advanced details**, expand the section to view the fields.
8. For **IAM instance profile**, choose the IAM instance profile name of your IAM role (**TestAutoScalingEvent-role**). An instance profile is a container for an IAM role that allows Amazon EC2 to pass the IAM role to an instance when the instance is launched.

When you used the IAM console to create an IAM role, the console automatically created an instance profile with the same name as its corresponding role.

9. For **User data**, copy and paste the following sample user data script into the field. Replace the sample text for `group_name` with the name of the Auto Scaling group that you want to create and `region` with the AWS Region you want your Auto Scaling group to use.

```
#!/bin/bash

function get_target_state {
    echo $(curl -s http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state)
}

function get_instance_id {
    echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
    region='us-west-2'

    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
        --lifecycle-hook-name TestAutoScalingEvent-hook \
        --auto-scaling-group-name $group_name \
        --lifecycle-action-result CONTINUE \
        --instance-id $instance_id \
        --region $region)
}

function main {
    while true
    do
        target_state=$(get_target_state)
        if [ \"$target_state\" = \"InService\" ]; then
            # Change hostname
            export new_hostname="${group_name}-$instance_id"
            hostname $new_hostname
            # Send callback
            complete_lifecycle_action
            break
        fi
        echo $target_state
        sleep 5
    done
}

main
```

This simple user data script does the following:

- Calls the instance metadata to retrieve the target lifecycle state and instance ID from the instance metadata

- Retrieves the target lifecycle state repeatedly until it changes to **InService**
- Changes the hostname of the instance to the instance ID prepended with the name of the Auto Scaling group, if the target lifecycle state is **InService**
- Sends a callback by calling the **complete-lifecycle-action** CLI command to signal Amazon EC2 Auto Scaling to **CONTINUE** the EC2 launch process

10. Choose **Create launch template**.

11. On the confirmation page, choose **Create Auto Scaling group**.

Note

For other examples that you can use as a reference for developing your user data script, see the [GitHub repository](#) for Amazon EC2 Auto Scaling.

Step 3: Create an Auto Scaling group

After you create your launch template, create an Auto Scaling group.

To create an Auto Scaling group

1. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group (**TestAutoScalingEvent-group**).
2. Choose **Next** to go to the **Choose instance launch options** page.
3. For **Network**, choose a VPC.
4. For **Availability Zones and subnets**, choose one or more subnets from one or more Availability Zones.
5. In the **Instance type requirements** section, use the default setting to simplify this step. (Do not override the launch template.) For this tutorial, you will launch only one On-Demand Instance using the instance type specified in your launch template.
6. Choose **Skip to review** at the bottom of the screen.
7. On the **Review** page, review the details of your Auto Scaling group, and then choose **Create Auto Scaling group**.

Step 4: Add a lifecycle hook

Add a lifecycle hook to hold the instance in a wait state until your lifecycle action is complete.

To add a lifecycle hook

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom part of the **Auto Scaling groups** page.

3. In the lower pane, on the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.
4. To define a lifecycle hook, do the following:
 - a. For **Lifecycle hook name**, enter **TestAutoScalingEvent-hook**.
 - b. For **Lifecycle transition**, choose **Instance launch**.
 - c. For **Heartbeat timeout**, enter **300** for the number of seconds to wait for a callback from your user data script.
 - d. For **Default result**, choose **ABANDON**. If the hook times out without receiving a callback from your user data script, the Auto Scaling group terminates the new instance.

- e. (Optional) Keep **Notification metadata** blank.
5. Choose **Create**.

Step 5: Test and verify the functionality

To test the functionality, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. The user data script runs and starts to check the instance's target lifecycle state soon after the instance launches. The script changes the hostname and sends a callback action when the target lifecycle state is `InService`. This usually takes only a few seconds to finish.

To increase the size of the Auto Scaling group

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group. View details in a lower pane while still seeing the top rows of the upper pane.
3. In the lower pane, on the **Details** tab, choose **Group details, Edit**.
4. For **Desired capacity**, increase the current value by 1.
5. Choose **Update**. While the instance is being launched, the **Status** column in the upper pane displays a status of *Updating capacity*.

After increasing the desired capacity, you can verify that your instance has successfully launched and is not terminated from the description of scaling activities.

To view the scaling activity

1. Return to the [Auto Scaling groups page](#) and select your group.
2. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched an instance.
3. If the user data script fails, after the timeout period passes, you see a scaling activity with a status of `Canceled` and a status message of `Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result`.

Step 6: Clean up

If you are done working with the resources that you created for this tutorial, use the following steps to delete them.

To delete the lifecycle hook

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.
3. On the **Instance management** tab, in **Lifecycle hooks**, choose the lifecycle hook (`TestAutoScalingEvent-hook`).
4. Choose **Actions, Delete**.
5. Choose **Delete** again to confirm.

To delete the launch template

1. Open the [Launch templates page](#) on the Amazon EC2 console.

2. Select your launch template (TestAutoScalingEvent-template) and then choose **Actions**, **Delete template**.
3. When prompted for confirmation, type **Delete** to confirm deleting the specified launch template and then choose **Delete**.

If you are done working with the example Auto Scaling group, delete it. You can also delete the IAM role and permissions policy that you created.

To delete the Auto Scaling group

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group (TestAutoScalingEvent-group) and choose **Delete**.
3. When prompted for confirmation, type **delete** to confirm deleting the specified Auto Scaling group and then choose **Delete**.

A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. It takes a few minutes to terminate the instances and delete the group.

To delete the IAM role

1. Open the [Roles page](#) of the IAM console.
2. Select the function's role (TestAutoScalingEvent-role).
3. Choose **Delete**.
4. When prompted for confirmation, type the name of the role and then choose **Delete**.

To delete the IAM policy

1. Open the [Policies page](#) of the IAM console.
2. Select the policy that you created (TestAutoScalingEvent-policy).
3. Choose **Actions, Delete**.
4. When prompted for confirmation, type the name of the policy and then choose **Delete**.

Tutorial: Configure a lifecycle hook that invokes a Lambda function

In this exercise, you create an Amazon EventBridge rule that includes a filter pattern that when matched, invokes an AWS Lambda function as the rule target. We provide the filter pattern and sample function code to use.

If everything is configured correctly, at the end of this tutorial, the Lambda function performs a custom action when instances launch. The custom action simply logs the event in the CloudWatch Logs log stream associated with the Lambda function.

The Lambda function also performs a callback to let the lifecycle of the instance proceed if this action is successful, but lets the instance abandon the launch and terminate if the action fails.

For more information about using EventBridge, see [Use EventBridge to handle Auto Scaling events \(p. 304\)](#).

Contents

- [Prerequisites \(p. 214\)](#)
- [Step 1: Create an IAM role with permissions to complete lifecycle actions \(p. 214\)](#)
- [Step 2: Create a Lambda function \(p. 215\)](#)
- [Step 3: Create an EventBridge rule \(p. 216\)](#)
- [Step 4: Add a lifecycle hook \(p. 216\)](#)
- [Step 5: Test and verify the event \(p. 217\)](#)
- [Step 6: Next steps \(p. 218\)](#)
- [Step 7: Clean up \(p. 218\)](#)

Prerequisites

Before you begin this tutorial, create an Auto Scaling group, if you don't have one already. To create an Auto Scaling group, open the [Auto Scaling groups page](#) of the Amazon EC2 console and choose **Create Auto Scaling group**.

Step 1: Create an IAM role with permissions to complete lifecycle actions

Before you create a Lambda function, you must first create an execution role and a permissions policy to allow Lambda to complete lifecycle hooks.

To create the policy

1. Open the [Policies page](#) of the IAM console, and then choose **Create policy**.
2. Choose the **JSON** tab.
3. In the **Policy Document** box, paste the following policy document into the box, replacing the text in *italics* with your account number and the name of your Auto Scaling group.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "autoscaling:CompleteLifecycleAction"  
      ],  
      "Resource":  
        "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-asg"  
    }  
  ]  
}
```

4. Choose **Next:Tags**, and then **Next:Review**.
5. For **Name**, enter **LogAutoScalingEvent-policy**. Choose **Create policy**.

When you finish creating the policy, you can create a role that uses it.

To create the role

1. In the navigation pane on the left, choose **Roles**.
2. Choose **Create role**.

3. For **Select trusted entity**, choose **AWS service**.
4. For your use case, choose **Lambda** and then choose **Next**.
5. Under **Add permissions**, choose the policy that you created (**LogAutoScalingEvent-policy**) and the policy named **AWSLambdaBasicExecutionRole**. Then, choose **Next**.

Note

The **AWSLambdaBasicExecutionRole** policy has the permissions that the function needs to write logs to CloudWatch Logs.

6. On the **Name, review, and create** page, for **Role name**, enter **LogAutoScalingEvent-role** and choose **Create role**.

Step 2: Create a Lambda function

Create a Lambda function to serve as the target for events. The sample Lambda function, written in `Node.js`, is invoked by EventBridge when a matching event is emitted by Amazon EC2 Auto Scaling.

To create a Lambda function

1. Open the [Functions page](#) on the Lambda console.
2. Choose **Create function, Author from scratch**.
3. Under **Basic information**, for **Function name**, enter **LogAutoScalingEvent**.
4. Choose **Change default execution role**, and then for **Execution role**, choose **Use an existing role**.
5. For **Existing role**, choose **LogAutoScalingEvent-role**.
6. Leave the other default values.
7. Choose **Create function**. You are returned to the function's code and configuration.
8. With your `LogAutoScalingEvent` function still open in the console, under **Function code**, in the editor, paste the following sample code into the file named `index.js`.

```
var aws = require("aws-sdk");
exports.handler = (event, context, callback) => {
    console.log('LogAutoScalingEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
    var autoscaling = new aws.AutoScaling({region: event.region});
    var eventDetail = event.detail;
    var params = {
        AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
        LifecycleActionResult: 'CONTINUE', /* required */
        LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
        InstanceId: eventDetail['EC2InstanceId'],
        LifecycleActionToken: eventDetail['LifecycleActionToken']
    };
    var response;
    autoscaling.completeLifecycleAction(params, function(err, data) {
        if (err) {
            console.log(err, err.stack); // an error occurred
            response = {
                statusCode: 500,
                body: JSON.stringify('ERROR'),
            };
        } else {
            console.log(data); // successful response
            response = {
                statusCode: 200,
                body: JSON.stringify('SUCCESS'),
            };
        }
    });
}
```

```
});  
return response;  
};
```

This code simply logs the event so that, at the end of this tutorial, you can see an event appear in the CloudWatch Logs log stream that's associated with this Lambda function.

9. Choose **Deploy**.

Step 3: Create an EventBridge rule

Create an EventBridge rule to run your Lambda function.

To create a rule using the console

1. Open the [EventBridge console](#).
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. For **Define rule detail**, do the following:
 - a. For **Name**, enter **LogAutoScalingEvent-rule**.
 - b. For **Event bus**, choose **default**. When an AWS service in your account generates an event, it always goes to your account's default event bus.
 - c. For **Rule type**, choose **Rule with an event pattern**.
 - d. Choose **Next**.
5. For **Build event pattern**, do the following:
 - a. For **Event source**, choose **AWS events or EventBridge partner events**.
 - b. For **Event pattern**, do the following:
 - i. For **Event source**, choose **AWS services**.
 - ii. For **AWS service**, choose **Auto Scaling**.
 - iii. For **Event type**, choose **Instance Launch and Terminate**.
 - iv. By default, the rule matches any scale-in or scale-out event. To create a rule that notifies you when there is a scale-out event and an instance is put into a wait state due to a lifecycle hook, choose **Specific instance event(s)** and select **EC2 Instance-launch Lifecycle Action**.
 - v. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific Auto Scaling group, choose **Specific group name(s)** and select one or more Auto Scaling groups.
 - vi. Choose **Next**.
6. For **Select target(s)**, do the following:
 - a. For **Target types**, choose **AWS service**.
 - b. For **Select a target**, choose **Lambda function**.
 - c. For **Function**, choose **LogAutoScalingEvent**.
 - d. Choose **Next** twice.
7. On the **Review and create** page, choose **Create**.

Step 4: Add a lifecycle hook

In this section, you add a lifecycle hook so that Lambda runs your function on instances at launch.

To add a lifecycle hook

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. In the lower pane, on the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.
4. To define a lifecycle hook, do the following:
 - a. For **Lifecycle hook name**, enter **LogAutoScalingEvent-hook**.
 - b. For **Lifecycle transition**, choose **Instance launch**.
 - c. For **Heartbeat timeout**, enter **300** for the number of seconds to wait for a callback from your Lambda function.
 - d. For **Default result**, choose **ABANDON**. This means that the Auto Scaling group will terminate a new instance if the hook times out without receiving a callback from your Lambda function.
 - e. (Optional) Leave **Notification metadata** empty. The event data that we pass to EventBridge contains all of the necessary information to invoke the Lambda function.
5. Choose **Create**.

Step 5: Test and verify the event

To test the event, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. Your Lambda function is invoked within a few seconds after increasing the desired capacity.

To increase the size of the Auto Scaling group

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group to view details in a lower pane and still see the top rows of the upper pane.
3. In the lower pane, on the **Details** tab, choose **Group details, Edit**.
4. For **Desired capacity**, increase the current value by 1.
5. Choose **Update**. While the instance is being launched, the **Status** column in the upper pane displays a status of *Updating capacity*.

After increasing the desired capacity, you can verify that your Lambda function was invoked.

To view the output from your Lambda function

1. Open the [Log groups page](#) of the CloudWatch console.
2. Select the name of the log group for your Lambda function (/aws/lambda/**LogAutoScalingEvent**).
3. Select the name of the log stream to view the data provided by the function for the lifecycle action.

Next, you can verify that your instance has successfully launched from the description of scaling activities.

To view the scaling activity

1. Return to the [Auto Scaling groups page](#) and select your group.

2. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched an instance.
 - If the action was successful, the scaling activity will have a status of "Successful".
 - If it failed, after waiting a few minutes, you will see a scaling activity with a status of "Cancelled" and a status message of "Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result".

To decrease the size of the Auto Scaling group

If you do not need the additional instance that you launched for this test, you can open the **Details** tab and decrease **Desired capacity** by 1.

Step 6: Next steps

Now that you have completed this tutorial, you can try creating a termination lifecycle hook. If instances in the Auto Scaling group terminate, an event is sent to EventBridge. For information about the event that is emitted when an instance terminates, see [EC2 Instance-terminate Lifecycle Action \(p. 306\)](#).

Step 7: Clean up

If you are done working with the resources that you created just for this tutorial, use the following steps to delete them.

To delete the lifecycle hook

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.
3. On the **Instance management** tab, in **Lifecycle hooks**, choose the lifecycle hook (LogAutoScalingEvent-hook).
4. Choose **Actions, Delete**.
5. Choose **Delete** again to confirm.

To delete the Amazon EventBridge rule

1. Open the [Rules page](#) in the Amazon EventBridge console.
2. Under **Event bus**, choose the event bus that is associated with the rule (Default).
3. Select the check box next to your rule (LogAutoScalingEvent-rule).
4. Choose **Delete**.
5. When prompted for confirmation, type the name of the rule and then choose **Delete**.

If you are done working with the example function, delete it. You can also delete the log group that stores the function's logs, and the execution role and permissions policy that you created.

To delete a Lambda function

1. Open the [Functions page](#) on the Lambda console.
2. Choose the function (LogAutoScalingEvent).
3. Choose **Actions, Delete**.

4. When prompted for confirmation, type **delete** to confirm deleting the specified function and then choose **Delete**.

To delete the log group

1. Open the [Log groups page](#) of the CloudWatch console.
2. Select the function's log group (/aws/lambda/LogAutoScalingEvent).
3. Choose **Actions, Delete log group(s)**.
4. In the **Delete log group(s)** dialog box, choose **Delete**.

To delete the execution role

1. Open the [Roles page](#) of the IAM console.
2. Select the function's role (LogAutoScalingEvent-role).
3. Choose **Delete**.
4. When prompted for confirmation, type the name of the role and then choose **Delete**.

To delete the IAM policy

1. Open the [Policies page](#) of the IAM console.
2. Select the policy that you created (LogAutoScalingEvent-policy).
3. Choose **Actions, Delete**.
4. When prompted for confirmation, type the name of the policy and then choose **Delete**.

Warm pools for Amazon EC2 Auto Scaling

A warm pool gives you the ability to decrease latency for your applications that have exceptionally long boot times, for example, because instances need to write massive amounts of data to disk. With warm pools, you no longer have to over-provision your Auto Scaling groups to manage latency in order to improve application performance. For more information, see the following blog post [Scaling your applications faster with EC2 Auto Scaling Warm Pools](#).

Important

Creating a warm pool when it's not required can lead to unnecessary costs. If your first boot time does not cause noticeable latency issues for your application, there probably isn't a need for you to use a warm pool.

Contents

- [Core concepts \(p. 220\)](#)
- [Prerequisites \(p. 221\)](#)
- [Create a warm pool \(p. 222\)](#)
- [Update a warm pool \(p. 222\)](#)
- [Delete a warm pool \(p. 223\)](#)
- [Limitations \(p. 223\)](#)
- [Use lifecycle hooks with a warm pool \(p. 223\)](#)
- [View health check status and the reason for health check failures \(p. 226\)](#)
- [Examples for creating and managing warm pools with the AWS CLI \(p. 228\)](#)

Core concepts

Before you get started, familiarize yourself with the following core concepts:

Warm pool

A warm pool is a pool of pre-initialized EC2 instances that sits alongside an Auto Scaling group. Whenever your application needs to scale out, the Auto Scaling group can draw on the warm pool to meet its new desired capacity. This helps you to ensure that instances are ready to quickly start serving application traffic, accelerating the response to a scale-out event. As instances leave the warm pool, they count toward the desired capacity of the group. This is known as a *warm start*.

While instances are in the warm pool, your scaling policies only scale out if the metric value from instances that are in the `InService` state is greater than the scaling policy's alarm high threshold (which is the same as the target utilization of a target tracking scaling policy).

Warm pool size

By default, the size of the warm pool is calculated as the difference between the Auto Scaling group's maximum capacity and its desired capacity. For example, if the desired capacity of your Auto Scaling group is 6 and the maximum capacity is 10, the size of your warm pool will be 4 when you first set up the warm pool and the pool is initializing.

To specify the warm pool's maximum capacity separately, set a value for maximum prepared capacity that is greater than the current capacity of the group. When you set a value for maximum prepared capacity, the size of the warm pool is calculated as the difference between the maximum prepared capacity and the current desired capacity of the group. For example, if the desired capacity of your Auto Scaling group is 6, if the maximum capacity is 10, and if the maximum prepared capacity is 8, the size of your warm pool will be 2 when you first set up the warm pool and the pool is initializing.

You might only need to use the maximum prepared capacity option when working with large Auto Scaling groups to manage the cost benefits of having a warm pool. For example, an Auto Scaling group with 1,000 instances, a maximum capacity of 1,500 (to provide extra capacity for emergency traffic spikes), and a warm pool of 100 instances might help you achieve your goals better than keeping 500 instances reserved for future use inside the warm pool.

Minimum warm pool size

Consider using the minimum size setting to statically set the minimum number of instances to maintain in the warm pool. There is no minimum size set by default.

Warm pool instance state

You can keep instances in the warm pool in one of three states: `Stopped`, `Running`, or `Hibernated`. Keeping instances in a `Stopped` state is an effective way to minimize costs. With stopped instances, you pay only for the volumes that you use and the Elastic IP addresses attached to the instances.

Alternatively, you can keep instances in a `Hibernated` state to stop instances without deleting their memory contents (RAM). When an instance is hibernated, this signals the operating system to save the contents of your RAM to your Amazon EBS root volume. When the instance is started again, the root volume is restored to its previous state and the RAM contents are reloaded. While the instances are in hibernation, you pay only for the EBS volumes, including storage for the RAM contents, and the Elastic IP addresses attached to the instances.

Keeping instances in a `Running` state inside the warm pool is also possible, but is highly discouraged to avoid incurring unnecessary charges. When instances are stopped or hibernated, you are saving the cost of the instances themselves. You pay for the instances only when they are running.

Lifecycle hooks

[Lifecycle hooks \(p. 195\)](#) let you put instances into a wait state so that you can perform custom actions on the instances. Custom actions are performed as the instances launch or before they terminate.

In a warm pool configuration, lifecycle hooks can also delay instances from being stopped or hibernated and from being put in service during a scale-out event until they have finished initializing. If you add a warm pool to your Auto Scaling group without a lifecycle hook, instances that take a long time to finish initializing could be stopped or hibernated and then put in service during a scale-out event before they are ready.

Instance reuse policy

By default, Amazon EC2 Auto Scaling terminates your instances when your Auto Scaling group scales in. Then, it launches new instances into the warm pool to replace the instances that were terminated.

If you want to return instances to the warm pool instead, you can specify an instance reuse policy. This lets you reuse instances that are already configured to serve application traffic. To make sure that your warm pool is not over-provisioned, Amazon EC2 Auto Scaling can terminate instances in the warm pool to reduce its size when it is larger than necessary based on its settings. When terminating instances in the warm pool, it uses the [default termination policy \(p. 233\)](#) to choose which instances to terminate first.

Important

If you want to hibernate instances on scale in and there are existing instances in the Auto Scaling group, they must meet the requirements for instance hibernation. If they don't, when instances return to the warm pool, they will fallback to being stopped instead of being hibernated.

Note

Currently, you can only specify an instance reuse policy by using the AWS CLI or an SDK. This feature is not available from the console.

Prerequisites

Decide how you will use lifecycle hooks to prepare the instances for use. There are two ways to perform custom actions on your instances.

- For simple scenarios where you want to run commands on your instances at launch, you can include a user data script when you create a launch template or launch configuration for your Auto Scaling group. User data scripts are just normal shell scripts or cloud-init directives that are run by [cloud-init](#) when your instances start. The script can also control when your instances transition to the next state by using the ID of the instance on which it runs. If you are not doing so already, update your script to retrieve the instance ID of the instance from the instance metadata. For more information, see [Retrieve instance metadata](#) in the *Amazon EC2 User Guide for Linux Instances*.

Tip

To run user data scripts when an instance restarts, the user data must be in the MIME multi-part format and specify the following in the `#cloud-config` section of the user data:

```
#cloud-config
cloud_final_modules:
  - [scripts-user, always]
```

- For advanced scenarios where you need a service such as AWS Lambda to do something as instances are entering or leaving the warm pool, you can create a lifecycle hook for your Auto Scaling group

and configure the target service to perform custom actions based on lifecycle notifications. For more information, see [Supported notification targets \(p. 225\)](#).

For more information, see the lifecycle hook examples in our [GitHub repository](#).

Prepare instances for hibernation

To prepare Auto Scaling instances to use the Hibernated pool state, create a new launch template or launch configuration that is set up correctly to support instance hibernation, as described in the [Hibernation prerequisites](#) topic in the *Amazon EC2 User Guide for Linux Instances*. Then, associate the new launch template or launch configuration with the Auto Scaling group and start an instance refresh to replace the instances associated with a previous launch template or launch configuration. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Create a warm pool

Create a warm pool using the console according to the following instructions.

Before you begin, confirm that you have created a lifecycle hook for your Auto Scaling group.

To create a warm pool (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.

A split pane opens up at the bottom of the **Auto Scaling groups** page.
3. Choose the **Instance management** tab.
4. Under **Warm pool**, choose **Create warm pool**.
5. To configure a warm pool, do the following:
 - a. For **Warm pool instance state**, choose which state you want to transition your instances to when they enter the warm pool. The default is **Stopped**.
 - b. For **Minimum warm pool size**, enter the minimum number of instances to maintain in the warm pool.
 - c. For **Max prepared capacity**, you can specify the maximum prepared capacity by defining a set number of instances, or keep the default option to keep the maximum prepared capacity undefined.

If you keep the default, **Equal to the Auto Scaling group's maximum capacity**, the warm pool size is sized to match the difference between the Auto Scaling group's maximum capacity and its desired capacity. To make it easier to manage the warm pool's size by adjusting the group's maximum capacity, we recommend that you use the default option.

If you choose the **Define a set number of instances** option, enter a value that represents the maximum number of instances that are allowed to be in the warm pool and the Auto Scaling group at the same time.
6. Choose **Create**.

Update a warm pool

To change the launch template or launch configuration for a warm pool, associate a new launch template or launch configuration with the Auto Scaling group. Any new instances are launched using the new AMI and other updates that are specified in the launch template or launch configuration, but existing instances are not affected.

To force replacement warm pool instances to launch that use the new launch template or launch configuration, you can terminate existing instances in the warm pool. Amazon EC2 Auto Scaling immediately starts launching new instances to replace the instances that you terminated. Alternatively, you can start an instance refresh to do a rolling update of your group. An instance refresh first replaces InService instances. Then it replaces instances in the warm pool. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Delete a warm pool

When you no longer need the warm pool, use the following procedure to delete it.

To delete your warm pool (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.
A split pane opens up at the bottom of the **Auto Scaling groups** page.
3. Choose the **Instance management** tab.
4. For **Warm pool**, choose **Actions, Delete**.
5. When prompted for confirmation, choose **Delete**.

Limitations

- You cannot add a warm pool to Auto Scaling groups that have a mixed instances policy or that launch Spot Instances.
- Amazon EC2 Auto Scaling can put an instance in a Stopped or Hibernated state only if it has an Amazon EBS volume as its root device. Instances that use instance stores for the root device cannot be stopped or hibernated.
- Amazon EC2 Auto Scaling can put an instance in a Hibernated state only if meets all of the requirements listed in the [Hibernation prerequisites](#) topic in the *Amazon EC2 User Guide for Linux Instances*.
- If your warm pool is depleted when there is a scale-out event, instances will launch directly into the Auto Scaling group (a *cold start*). You could also experience cold starts if an Availability Zone is out of capacity.
- If you try using a warm pool with an Amazon Elastic Kubernetes Service (Amazon EKS) managed node group, instances that are still initializing might register with your Amazon EKS cluster. As a result, the cluster might schedule jobs on an instance as it is preparing to be stopped or hibernated.
- Likewise, if you try using a warm pool with an Amazon ECS cluster, instances might register with the cluster before they finish initializing. To solve this problem, you must configure a launch template or launch configuration that includes a special agent configuration variable in the user data. For more information, see [Using a warm pool for your Auto Scaling group](#) in the *Amazon Elastic Container Service Developer Guide*.
- Hibernation support for warm pools is available in all commercial AWS Regions where Amazon EC2 Auto Scaling is available, excluding the Middle East (UAE), China (Beijing), China (Ningxia), and AWS GovCloud (US-East and US-West) Regions.

Use lifecycle hooks with a warm pool

Instances in a warm pool maintain their own independent lifecycle to help you create the appropriate custom action for each transition. This lifecycle is designed to help you to invoke actions in a target

service (for example, a Lambda function) while an instance is still initializing and before it is put in service.

Note

The API operations that you use to add and manage lifecycle hooks and complete lifecycle actions are not changed. Only the instance lifecycle is changed.

For more information about adding a lifecycle hook, see [Add lifecycle hooks \(p. 205\)](#). For more information about completing a lifecycle action, see [Complete a lifecycle action \(p. 207\)](#).

For instances entering the warm pool, you might need a lifecycle hook for one of the following reasons:

- You want to launch EC2 instances from an AMI that takes a long time to finish initializing.
- You want to run user data scripts to bootstrap the EC2 instances.

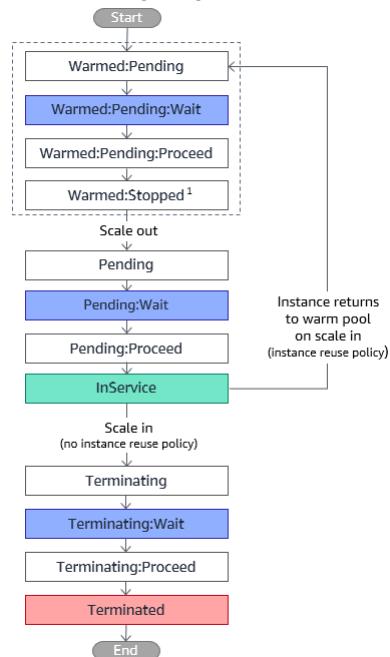
For instances leaving the warm pool, you might need a lifecycle hook for one of the following reasons:

- You can use some extra time to prepare EC2 instances for use. For example, you might have services that must start when an instance restarts before your application can work correctly.
- You want to pre-populate cache data so that a new server doesn't launch with an empty cache.
- You want to register new instances as managed instances with your configuration management service.

Lifecycle state transitions for instances in a warm pool

An Auto Scaling instance can transition through many states as part of its lifecycle.

The following diagram shows the transition between Auto Scaling states when you use a warm pool:



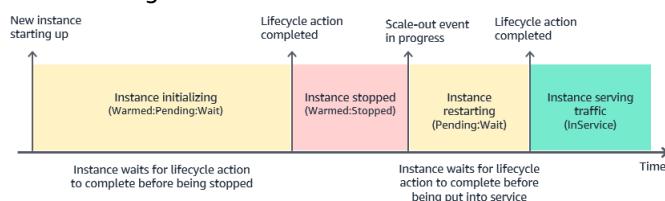
Instance returns to warm pool on scale in (instance reuse policy)

¹ This state varies based on the warm pool's pool state setting. If the pool state is set to Running, then this state is Warmed:Running instead. If the pool state is set to Hibernated, then this state is Warmed:Hibernated instead.

When you add lifecycle hooks, consider the following:

- When a lifecycle hook is configured for the `autoscaling:EC2_INSTANCE_LAUNCHING` lifecycle action, a newly launched instance first pauses to perform a custom action when it reaches the `Warmed:Pending:Wait` state, and then again when the instance restarts and reaches the `Pending:Wait` state.
- When a lifecycle hook is configured for the `EC2_INSTANCE_TERMINATING` lifecycle action, a terminating instance pauses to perform a custom action when it reaches the `Terminating:Wait` state. However, if you specify an instance reuse policy to return instances to the warm pool on scale in instead of terminating them, then an instance that is returning to the warm pool pauses to perform a custom action at the `Warmed:Pending:Wait` state for the `EC2_INSTANCE_TERMINATING` lifecycle action.
- If the demand on your application depletes the warm pool, Amazon EC2 Auto Scaling can launch instances directly into the Auto Scaling group as long as the group isn't at its maximum capacity yet. If the instances launch directly into the group, they are only paused to perform a custom action at the `Pending:Wait` state.
- To control how long an instance stays in a wait state before it transitions to the next state, configure your custom action to use the **complete-lifecycle-action** command. With lifecycle hooks, instances remain in a wait state either until you notify Amazon EC2 Auto Scaling that the specified lifecycle action is complete, or until the timeout period ends (one hour by default).

The following summarizes the flow for a scale-out event.



When instances reach a wait state, Amazon EC2 Auto Scaling sends a notification. Examples of these notifications are available in the EventBridge section of this guide. For more information, see [Warm pool event types and patterns \(p. 309\)](#).

Supported notification targets

Amazon EC2 Auto Scaling provides support for defining any of the following as notification targets for lifecycle notifications:

- EventBridge rules
- Amazon SNS topics
- Amazon SQS queues

Important

Remember, if you have a user data (cloud-init) script in your launch template or launch configuration that configures your instances when they launch, you do not need to receive notifications to perform custom actions on instances that are starting or restarting.

The following sections contain links to documentation that describes how to configure notification targets:

EventBridge rules: To run code when Amazon EC2 Auto Scaling puts an instance into a wait state, you can create an EventBridge rule and specify a Lambda function as its target. To invoke different Lambda functions based on different lifecycle notifications, you can create multiple rules and associate each rule with a specific event pattern and Lambda function. For more information, see [Create EventBridge rules for warm pool events \(p. 316\)](#).

Amazon SNS topics: To receive a notification when an instance is put into a wait state, you create an Amazon SNS topic and then set up Amazon SNS message filtering to deliver lifecycle notifications differently based on a message attribute. For more information, see [Receive notifications using Amazon SNS \(p. 202\)](#).

Amazon SQS queues: To set up a delivery point for lifecycle notifications where a relevant consumer can pick them up and process them, you can create an Amazon SQS queue and a queue consumer that processes messages from the SQS queue. If you want the queue consumer to process lifecycle notifications differently based on a message attribute, you must also set up the queue consumer to parse the message and then act on the message when a specific attribute matches the desired value. For more information, see [Receive notifications using Amazon SQS \(p. 202\)](#).

View health check status and the reason for health check failures

Health checks allow Amazon EC2 Auto Scaling to determine when an instance is unhealthy and should be terminated. For warm pool instances kept in a Stopped state, it employs the knowledge that Amazon EBS has of a Stopped instance's availability to identify unhealthy instances. It does this by calling the `DescribeVolumeStatus` API to determine the status of the EBS volume that's attached to the instance. For warm pool instances kept in a Running state, it relies on EC2 status checks to determine instance health. While there is no health check grace period for warm pool instances, Amazon EC2 Auto Scaling doesn't start checking instance health until the lifecycle hook finishes.

When an instance is found to be unhealthy, Amazon EC2 Auto Scaling automatically deletes the unhealthy instance and creates a new one to replace it. Instances are usually terminated within a few minutes after failing their health check. For more information, see [Unhealthy instance replacement \(p. 256\)](#).

Custom health checks are also supported. This can be helpful if you have your own health check system that can detect an instance's health and send this information to Amazon EC2 Auto Scaling. For more information, see [Custom health detection tasks \(p. 255\)](#).

On the Amazon EC2 Auto Scaling console, you can view the status (healthy or unhealthy) of your warm pool instances. You can also view their health status using the AWS CLI or one of the SDKs.

To view the status of your warm pool instances (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Instance management** tab, under **Warm pool instances**, the **Lifecycle** column contains the state of your instances.

The **Health status** column shows the assessment that Amazon EC2 Auto Scaling has made of instance health.

Note

New instances start healthy. Until the lifecycle hook is finished, an instance's health is not checked.

To view the reason for health check failures (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.

2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances.

If it terminated any unhealthy instances, the **Cause** column shows the date and time of the termination and the reason for the health check failure. For example, "At 2021-04-01T21:48:35Z an instance was taken out of service in response to EBS volume health check failure".

To view the status of your warm pool instances (AWS CLI)

View the warm pool for an Auto Scaling group by using the following [describe-warm-pool](#) command.

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

Example output.

```
{  
  "WarmPoolConfiguration": {  
    "MinSize": 0,  
    "PoolState": "Stopped"  
  },  
  "Instances": [  
    {  
      "InstanceId": "i-0b5e5e7521cf8a46c",  
      "InstanceType": "t2.micro",  
      "AvailabilityZone": "us-west-2a",  
      "LifecycleState": "Warmed:Stopped",  
      "HealthStatus": "Healthy",  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-08c4cd42f320d5dcd",  
        "LaunchTemplateName": "my-template-for-auto-scaling",  
        "Version": "1"  
      }  
    },  
    {  
      "InstanceId": "i-0e21af9dcfb7aa6bf",  
      "InstanceType": "t2.micro",  
      "AvailabilityZone": "us-west-2a",  
      "LifecycleState": "Warmed:Stopped",  
      "HealthStatus": "Healthy",  
      "LaunchTemplate": {  
        "LaunchTemplateId": "lt-08c4cd42f320d5dcd",  
        "LaunchTemplateName": "my-template-for-auto-scaling",  
        "Version": "1"  
      }  
    }  
  ]  
}
```

To view the reason for health check failures (AWS CLI)

Use the following [describe-scaling-activities](#) command.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

The following is an example response, where **Description** indicates that your Auto Scaling group has terminated an instance and **Cause** indicates the reason for the health check failure.

Scaling activities are ordered by start time. Activities still in progress are described first.

```
{  
  "Activities": [  
    {  
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",  
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in response to  
EBS volume health check failure.",  
      "StartTime": "2021-04-01T21:48:35.859Z",  
      "EndTime": "2021-04-01T21:49:18Z",  
      "StatusCode": "Successful",  
      "Progress": 100,  
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\", \"Availability Zone\":\"us-west-2a\"}  
\\\"...\\\",  
      \"AutoScalingGroupARN\": \"arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:283179a2-  
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"  
    },  
    ...  
  ]  
}
```

Examples for creating and managing warm pools with the AWS CLI

You can create and manage warm pools using the AWS Management Console, AWS Command Line Interface (AWS CLI), or SDKs.

The following examples show you how to create and manage warm pools using the AWS CLI.

Contents

- [Example 1: Keep instances in the Stopped state \(p. 228\)](#)
- [Example 2: Keep instances in the Running state \(p. 228\)](#)
- [Example 3: Keep instances in the Hibernated state \(p. 229\)](#)
- [Example 4: Return instances to the warm pool when scaling in \(p. 229\)](#)
- [Example 5: Specify the minimum number of instances in the warm pool \(p. 229\)](#)
- [Example 6: Define the warm pool maximum capacity separately \(p. 229\)](#)
- [Example 7: Define an absolute warm pool size \(p. 230\)](#)
- [Example 8: Delete a warm pool \(p. 230\)](#)

Example 1: Keep instances in the Stopped state

The following `put-warm-pool` example creates a warm pool that keeps instances in a Stopped state.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped
```

Example 2: Keep instances in the Running state

The following `put-warm-pool` example creates a warm pool that keeps instances in a Running state instead of a Stopped state.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Running
```

Example 3: Keep instances in the Hibernated state

The following [put-warm-pool](#) example creates a warm pool that keeps instances in a Hibernated state instead of a Stopped state. This lets you stop instances without deleting their memory contents (RAM).

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Hibernated
```

Example 4: Return instances to the warm pool when scaling in

The following [put-warm-pool](#) example creates a warm pool that keeps instances in a Stopped state and includes the `--instance-reuse-policy` option. The instance reuse policy value `'{"ReuseOnScaleIn": true}'` tells Amazon EC2 Auto Scaling to return instances to the warm pool when your Auto Scaling group scales in.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

Example 5: Specify the minimum number of instances in the warm pool

The following [put-warm-pool](#) example creates a warm pool that maintains a minimum of 4 instances, so that there are at least 4 instances available to handle traffic spikes.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --min-size 4
```

Example 6: Define the warm pool maximum capacity separately

Generally, you understand how much to increase your maximum capacity above your desired capacity. There's usually no need to define an additional maximum size, because Amazon EC2 Auto Scaling creates a warm pool that dynamically resizes based on your group's desired and maximum capacity. However, you can use the `--max-group-prepared-capacity` option to define the warm pool maximum capacity separately when desired.

The following [put-warm-pool](#) example creates a warm pool that defines its maximum capacity separately. Suppose that the Auto Scaling group has a desired capacity of 800. The size of the warm pool will be 100 when you run this command and the pool is initializing.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --max-group-prepared-capacity 900
```

To maintain a minimum number of instances in the warm pool, include the `--min-size` option with the command, as follows.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
--pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

Example 7: Define an absolute warm pool size

If you set the same values for the `--max-group-prepared-capacity` and `--min-size` options, the warm pool has an absolute size. The following `put-warm-pool` example creates a warm pool that maintains a constant warm pool size of 10 instances.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /  
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

Example 8: Delete a warm pool

Use the following `delete-warm-pool` command to delete a warm pool.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

If there are instances in the warm pool, or if scaling activities are in progress, use the `delete-warm-pool` command with the `--force-delete` option. This option also terminates the Amazon EC2 instances and any outstanding lifecycle actions.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Control which Auto Scaling instances terminate during scale in

Amazon EC2 Auto Scaling uses termination policies to determine which instances it terminates first during scale-in events. Termination policies define the termination criteria that is used by Amazon EC2 Auto Scaling when choosing which instances to terminate.

Your Auto Scaling groups use a default termination policy, but you can optionally choose or create your own termination policies with your own termination criteria. This lets you ensure that your instances are terminated based on your specific application needs.

Amazon EC2 Auto Scaling also provides instance scale-in protection. When you enable this feature, it prevents instances from being terminated during scale-in events. You can enable instance scale-in protection when you create an Auto Scaling group, and you can change the setting on running instances. If you enable instance scale-in protection on an existing Auto Scaling group, all new instances launched after that will have instance scale-in protection enabled.

Note

Instance scale-in protection does not guarantee that instances won't be terminated in the event of a human error—for example, if someone manually terminates an instance using the Amazon EC2 console or AWS CLI. To protect your instance from accidental termination, you can use Amazon EC2 termination protection. However, even with termination protection and instance scale-in protection enabled, data saved to instance storage can be lost if a health check determines that an instance is unhealthy or if the group itself is accidentally deleted. As with any environment, a best practice is to back up your data frequently, or whenever it's appropriate for your business continuity requirements.

Contents

- [Scenarios for termination policy use \(p. 231\)](#)
- [Work with Amazon EC2 Auto Scaling termination policies \(p. 233\)](#)

- [Create a custom termination policy with Lambda \(p. 236\)](#)
- [Use instance scale-in protection \(p. 240\)](#)

Scenarios for termination policy use

The following sections describe the scenarios in which Amazon EC2 Auto Scaling uses termination policies.

Contents

- [Scale-in events \(p. 231\)](#)
- [Instance refreshes \(p. 232\)](#)
- [Availability Zone rebalancing \(p. 232\)](#)

Scale-in events

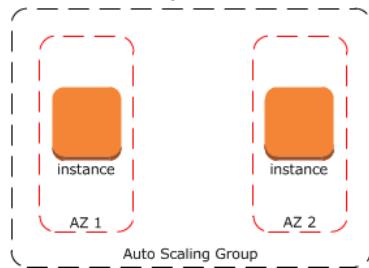
A scale-in event occurs when there is a new value for the desired capacity of an Auto Scaling group that is lower than the current capacity of the group.

Scale-in events occur in the following scenarios:

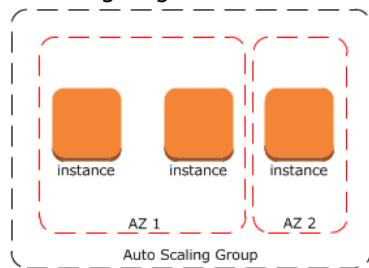
- When using dynamic scaling policies and the size of the group decreases as a result of changes in a metric's value
- When using scheduled scaling and the size of the group decreases as a result of a scheduled action
- When you manually decrease the size of the group

The following example shows how termination policies work when there is a scale-in event.

1. The Auto Scaling group in this example has one instance type, two Availability Zones, and a desired capacity of two instances. It also has a dynamic scaling policy that adds and removes instances when resource utilization increases or decreases. The two instances in this group are distributed across the two Availability Zones as shown in the following diagram.



2. When the Auto Scaling group scales out, Amazon EC2 Auto Scaling launches a new instance. The Auto Scaling group now has three instances, distributed across the two Availability Zones as shown in the following diagram.



3. When the Auto Scaling group scales in, Amazon EC2 Auto Scaling terminates one of the instances.
4. If you did not assign a specific termination policy to the group, Amazon EC2 Auto Scaling uses the default termination policy. It selects the Availability Zone with two instances, and terminates the instance that was launched from the oldest launch template or launch configuration. If the instances were launched from the same launch template or launch configuration, Amazon EC2 Auto Scaling selects the instance that is closest to the next billing hour and terminates it.

Instance refreshes

You start instance refreshes in order to update the instances in your Auto Scaling group. During an instance refresh, Amazon EC2 Auto Scaling terminates instances in the group and then launches replacements for the terminated instances. The termination policy for the Auto Scaling group controls which instances are replaced first.

Availability Zone rebalancing

Amazon EC2 Auto Scaling balances your capacity evenly across the Availability Zones enabled for your Auto Scaling group. This helps reduce the impact of an Availability Zone outage. If the distribution of capacity across Availability Zones becomes out of balance, Amazon EC2 Auto Scaling rebalances the Auto Scaling group by launching instances in the enabled Availability Zones with the fewest instances and terminating instances elsewhere. The termination policy controls which instances are prioritized for termination first.

There are a number of reasons why the distribution of instances across Availability Zones can become out of balance.

Removing instances

If you detach instances from your Auto Scaling group, you put instances on standby, or you explicitly terminate instances and decrement the desired capacity, which prevents replacement instances from launching, the group can become unbalanced. If this occurs, Amazon EC2 Auto Scaling compensates by rebalancing the Availability Zones.

Using different Availability Zones than originally specified

If you expand your Auto Scaling group to include additional Availability Zones, or you change which Availability Zones are used, Amazon EC2 Auto Scaling launches instances in the new Availability Zones and terminates instances in other zones to help ensure that your Auto Scaling group spans Availability Zones evenly.

Availability outage

Availability outages are rare. However, if one Availability Zone becomes unavailable and recovers later, your Auto Scaling group can become unbalanced between Availability Zones. Amazon EC2 Auto Scaling tries to gradually rebalance the group, and rebalancing might terminate instances in other zones.

For example, imagine that you have an Auto Scaling group that has one instance type, two Availability Zones, and a desired capacity of two instances. In a situation where one Availability Zone fails, Amazon EC2 Auto Scaling automatically launches a new instance in the healthy Availability Zone to replace the one in the unhealthy Availability Zone. Then, when the unhealthy Availability Zone returns to a healthy state later on, Amazon EC2 Auto Scaling automatically launches a new instance in this zone, which in turn terminates an instance in the unaffected zone.

Note

When rebalancing, Amazon EC2 Auto Scaling launches new instances before terminating the old ones, so that rebalancing does not compromise the performance or availability of your application.

Because Amazon EC2 Auto Scaling attempts to launch new instances before terminating the old ones, being at or near the specified maximum capacity could impede or completely stop rebalancing activities. To avoid this problem, the system can temporarily exceed the specified maximum capacity of a group by a 10 percent margin (or by a margin of one instance, whichever is greater) during a rebalancing activity. The margin is extended only if the group is at or near maximum capacity and needs rebalancing, either because of user-requested rezoning or to compensate for zone availability issues. The extension lasts only as long as needed to rebalance the group.

Work with Amazon EC2 Auto Scaling termination policies

This topic provides detailed information about the default termination policy and the options available to you to choose different termination policies for individual Auto Scaling groups. By choosing different termination policies, you can control which instances you prefer to terminate first when a scale-in event occurs. For example, you can choose a different termination policy so that Amazon EC2 Auto Scaling prioritizes terminating the oldest instances first.

When Amazon EC2 Auto Scaling terminates instances, it attempts to maintain balance across the Availability Zones that are used by your Auto Scaling group. Maintaining balance across Availability Zones takes precedence over termination policies. If one Availability Zone has more instances than the other Availability Zones that are used by the group, Amazon EC2 Auto Scaling applies the termination policies to the instances from the imbalanced Availability Zone. If the Availability Zones used by the group are balanced, Amazon EC2 Auto Scaling applies the termination policies across all of the Availability Zones for the group.

Contents

- [Default termination policy \(p. 233\)](#)
- [Default termination policy and mixed instances groups \(p. 234\)](#)
- [Use different termination policies \(p. 234\)](#)
 - [Use different termination policies \(console\) \(p. 235\)](#)
 - [Use different termination policies \(AWS CLI\) \(p. 235\)](#)

Default termination policy

The default termination policy applies multiple termination criteria before selecting an instance to terminate. When Amazon EC2 Auto Scaling terminates instances, it first determines which Availability Zones have the most instances, and it finds at least one instance that is not protected from scale in. Within the selected Availability Zone, the following default termination policy behavior applies:

1. Determine whether any of the instances eligible for termination use the oldest launch template or launch configuration:
 - a. [For Auto Scaling groups that use a launch template]

Determine whether any of the instances use the oldest launch template, unless there are instances that use a launch configuration. Amazon EC2 Auto Scaling terminates instances that use a launch configuration before it terminates instances that use a launch template.

- b. [For Auto Scaling groups that use a launch configuration]

Determine whether any of the instances use the oldest launch configuration.

2. After applying the preceding criteria, if there are multiple unprotected instances to terminate, determine which instances are closest to the next billing hour. If there are multiple unprotected instances closest to the next billing hour, terminate one of these instances at random.

Note that terminating the instance closest to the next billing hour helps you maximize the use of your instances that have an hourly charge. Alternatively, if your Auto Scaling group uses Amazon Linux, Windows, or Ubuntu, your EC2 usage is billed in one-second increments. For more information, see [Amazon EC2 pricing](#).

Default termination policy and mixed instances groups

When an Auto Scaling group with a [mixed instances policy \(p. 52\)](#) scales in, Amazon EC2 Auto Scaling still uses termination policies to prioritize which instances to terminate, but first it identifies which of the two types (Spot or On-Demand) should be terminated. It then applies the termination policies in each Availability Zone individually. It also identifies which instances (within the identified purchase option) in which Availability Zones to terminate that will result in the Availability Zones being most balanced. The same logic applies to Auto Scaling groups that use a mixed instances configuration with weights defined for the instance types.

The default termination policy changes slightly due to differences in how [mixed instances policies \(p. 52\)](#) are implemented. The following new behavior of the default termination policy applies:

1. Determine which instances are eligible for termination in order to align the remaining instances to the [allocation strategy \(p. 53\)](#) for the On-Demand or Spot Instance that is terminating.

For example, after your instances launch, you might change the priority order of your preferred instance types. When a scale-in event occurs, Amazon EC2 Auto Scaling tries to gradually shift the On-Demand Instances away from instance types that are lower priority.
2. Determine whether any of the instances use the oldest launch template, unless there are instances that use a launch configuration. Amazon EC2 Auto Scaling terminates instances that use a launch configuration before it terminates instances that use a launch template.
3. After applying the preceding criteria, if there are multiple unprotected instances to terminate, determine which instances are closest to the next billing hour. If there are multiple unprotected instances closest to the next billing hour, terminate one of these instances at random.

Use different termination policies

To specify the termination criteria to apply before Amazon EC2 Auto Scaling chooses an instance for termination, you can choose from any of the following predefined termination policies:

- **Default.** Terminate instances according to the default termination policy. This policy is useful when you want your Spot allocation strategy evaluated before any other policy, so that every time your Spot instances are terminated or replaced, you continue to make use of Spot Instances in the optimal pools. It is also useful, for example, when you want to move off launch configurations and start using launch templates.
- **AllocationStrategy.** Terminate instances in the Auto Scaling group to align the remaining instances to the allocation strategy for the type of instance that is terminating (either a Spot Instance or an On-Demand Instance). This policy is useful when your preferred instance types have changed. If the Spot allocation strategy is **lowest-price**, you can gradually rebalance the distribution of Spot Instances across your N lowest priced Spot pools. If the Spot allocation strategy is **capacity-optimized**, you can gradually rebalance the distribution of Spot Instances across Spot pools where there is more available Spot capacity. You can also gradually replace On-Demand Instances of a lower priority type with On-Demand Instances of a higher priority type.
- **OldestLaunchTemplate.** Terminate instances that have the oldest launch template. With this policy, instances that use the noncurrent launch template are terminated first, followed by instances that use the oldest version of the current launch template. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.

- `OldestLaunchConfiguration`. Terminate instances that have the oldest launch configuration. This policy is useful when you're updating a group and phasing out the instances from a previous configuration. With this policy, instances that use the noncurrent launch configuration are terminated first.
- `ClosestToNextInstanceHour`. Terminate instances that are closest to the next billing hour. This policy helps you maximize the use of your instances that have an hourly charge. (Only instances that use Amazon Linux, Windows, or Ubuntu are billed in one-second increments.)
- `NewestInstance`. Terminate the newest instance in the group. This policy is useful when you're testing a new launch configuration but don't want to keep it in production.
- `OldestInstance`. Terminate the oldest instance in the group. This option is useful when you're upgrading the instances in the Auto Scaling group to a new EC2 instance type. You can gradually replace instances of the old type with instances of the new type.

Note

Amazon EC2 Auto Scaling always balances instances across Availability Zones first, regardless of which termination policy is used. As a result, you might encounter situations in which some newer instances are terminated before older instances. For example, when there is a more recently added Availability Zone, or when one Availability Zone has more instances than the other Availability Zones that are used by the group.

Use different termination policies (console)

After your Auto Scaling group has been created, you can update the termination policies for your group. The default termination policy is used automatically. You have the option of replacing the default policy with a different termination policy (such as `OldestLaunchTemplate`) or multiple termination policies listed in the order in which they should apply.

To choose different termination policies

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Advanced configurations, Edit**.
4. For **Termination policies**, choose one or more termination policies. If you choose multiple policies, put them in the order that you want them evaluated in.

You can optionally choose **Custom termination policy** and then choose a Lambda function that meets your needs. If you have created versions and aliases for your Lambda function, you can choose a version or alias from the **Version/Alias** drop-down. To use the unpublished version of your Lambda function, keep **Version/Alias** set to its default. For more information, see [Create a custom termination policy with Lambda \(p. 236\)](#).

Note

When using multiple policies, their order must be set correctly:

- If you use the **Default** policy, it must be the last policy in the list.
- If you use a **Custom termination policy**, it must be the first policy in the list.

5. Choose **Update**.

Use different termination policies (AWS CLI)

The default termination policy is used automatically unless a different policy is specified.

To use a different termination policy

Use one of the following commands:

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

You can use termination policies individually, or combine them into a list of policies. For example, use the following command to update an Auto Scaling group to use the `OldestLaunchConfiguration` policy first and then use the `ClosestToNextInstanceHour` policy.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

If you use the `Default` termination policy, make it the last one in the list of termination policies. For example, `--termination-policies "OldestLaunchConfiguration" "Default"`.

To use a custom termination policy, you must first create your termination policy using AWS Lambda. To specify the Lambda function to use as your termination policy, make it the first one in the list of termination policies. For example, `--termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration"`. For more information, see [Create a custom termination policy with Lambda \(p. 236\)](#).

Create a custom termination policy with Lambda

Amazon EC2 Auto Scaling uses termination policies to prioritize which instances to terminate first when decreasing the size of your Auto Scaling group (referred to as *scaling in*). Your Auto Scaling group uses a default termination policy, but you can optionally choose or create your own termination policies. For more information about choosing a predefined termination policy, see [Work with Amazon EC2 Auto Scaling termination policies \(p. 233\)](#).

In this topic, you learn how to create a custom termination policy using an AWS Lambda function that Amazon EC2 Auto Scaling invokes in response to certain events. The Lambda function that you create processes the information in the input data sent by Amazon EC2 Auto Scaling and returns a list of instances that are ready to terminate.

A custom termination policy provides better control over which instances are terminated, and when. For example, when your Auto Scaling group scales in, Amazon EC2 Auto Scaling cannot determine whether there are workloads running that should not be disrupted. With a Lambda function, you can validate the termination request and wait until the workload is done before returning the instance ID to Amazon EC2 Auto Scaling for termination.

Contents

- [Input data \(p. 236\)](#)
- [Response data \(p. 238\)](#)
- [Considerations when using a custom termination policy \(p. 238\)](#)
- [Create the Lambda function \(p. 239\)](#)
- [Limitations \(p. 240\)](#)

Input data

Amazon EC2 Auto Scaling generates a JSON payload for scale-in events, and also does so when instances are about to be terminated as a result of the maximum instance lifetime or instance refresh features. It

also generates a JSON payload for the scale-in events that it can initiate when rebalancing your group across Availability Zones.

This payload contains information about the capacity Amazon EC2 Auto Scaling needs to terminate, a list of instances that it suggests for termination, and the event that initiated the termination.

The following is an example payload:

```
{
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 2,
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 1,
      "InstanceMarketOption": "spot"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "Capacity": 3,
      "InstanceMarketOption": "on-demand"
    }
  ],
  "Instances": [
    {
      "AvailabilityZone": "us-east-1b",
      "InstanceId": "i-0056faf8da3e1f75d",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-02e1c69383a3ed501",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-036bc44b6092c01c7",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    ...
  ],
  "Cause": "SCALE_IN"
}
```

The payload includes the name of the Auto Scaling group, its Amazon Resource Name (ARN), and the following elements:

- **CapacityToTerminate** describes how much of your Spot or On-Demand capacity is set to be terminated in a given Availability Zone.
- **Instances** represents the instances that Amazon EC2 Auto Scaling suggests for termination based on the information in **CapacityToTerminate**.
- **Cause** describes the event that caused the termination: **SCALE_IN**, **INSTANCE_REFRESH**, **MAX_INSTANCE_LIFETIME**, or **REBALANCE**.

The following information outlines the most significant factors in how Amazon EC2 Auto Scaling generates the Instances in the input data:

- Maintaining balance across Availability Zones takes precedence when an instance is terminating due to scale-in events and instance refresh-based terminations. Therefore, if one Availability Zone has more instances than the other Availability Zones that are used by the group, the input data contains instances that are eligible for termination only from the imbalanced Availability Zone. If the Availability Zones used by the group are balanced, the input data contains instances from all of the Availability Zones for the group.
- When using a [mixed instances policy \(p. 52\)](#), maintaining your Spot and On-Demand capacities in balance based on your desired percentages for each purchase option also takes precedence. We first identify which of the two types (Spot or On-Demand) should be terminated. We then identify which instances (within the identified purchase option) in which Availability Zones we can terminate that will result in the Availability Zones being most balanced.

Response data

The input data and response data work together to narrow down the list of instances to terminate.

With the given input, the response from your Lambda function should look like the following example:

```
{  
  "InstanceIDs": [  
    "i-02e1c69383a3ed501",  
    "i-036bc44b6092c01c7",  
    ...  
  ]  
}
```

The InstanceIDs in the response represent the instances that are ready to terminate.

Alternatively, you can return a different set of instances that are ready to be terminated, which overrides the instances in the input data. If no instances are ready to terminate when your Lambda function is invoked, you can also choose not to return any instances.

When no instances are ready to terminate, the response from your Lambda function should look like the following example:

```
{  
  "InstanceIDs": []  
}
```

Considerations when using a custom termination policy

Note the following considerations when using a custom termination policy:

- Returning an instance first in the response data does not guarantee its termination. If more than the required number of instances are returned when your Lambda function is invoked, Amazon EC2 Auto Scaling evaluates each instance against the other termination policies that you specified for your Auto Scaling group. When there are multiple termination policies, it tries to apply the next termination policy in the list, and if there are more instances than are required to terminate, it moves on to the next termination policy, and so on. If no other termination policies are specified, then the default termination policy is used to determine which instances to terminate.
- If no instances are returned or your Lambda function times out, then Amazon EC2 Auto Scaling waits a short time before invoking your function again. For any scale-in event, it keeps trying as long as the group's desired capacity is less than its current capacity. For instance refresh-based terminations,

it keeps trying for an hour. After that, if it continues to fail to terminate any instances, the instance refresh operation fails. With maximum instance lifetime, Amazon EC2 Auto Scaling keeps trying to terminate the instance that is identified as exceeding its maximum lifetime.

- Because your function is retried repeatedly, make sure to test and fix any permanent errors in your code before using a Lambda function as a custom termination policy.
- If you override the input data with your own list of instances to terminate, and terminating these instances puts the Availability Zones out of balance, Amazon EC2 Auto Scaling gradually rebalances the distribution of capacity across Availability Zones. First, it invokes your Lambda function to see if there are instances that are ready to be terminated so that it can determine whether to start rebalancing. If there are instances ready to be terminated, it launches new instances first. When the instances finish launching, it then detects that your group's current capacity is higher than its desired capacity and initiates a scale-in event.

Create the Lambda function

Start by creating the Lambda function, so that you can specify its Amazon Resource Name (ARN) in the termination policies for your Auto Scaling group.

To create a Lambda function (console)

1. Open the [Functions page](#) on the Lambda console.
2. On the navigation bar at the top of the screen, choose the same Region that you used when you created the Auto Scaling group.
3. Choose **Create function, Author from scratch**.
4. Under **Basic information**, for **Function name**, enter the name of your function.
5. Choose **Create function**. You are returned to the function's code and configuration.
6. With your function still open in the console, under **Function code**, paste your code into the editor.
7. Choose **Deploy**.
8. Optionally, create a published version of the Lambda function by choosing the **Versions** tab and then **Publish new version**. To learn more about versioning in Lambda, see [Lambda function versions](#) in the *AWS Lambda Developer Guide*.
9. If you chose to publish a version, choose the **Aliases** tab if you want to associate an alias with this version of the Lambda function. To learn more about aliases in Lambda, see [Lambda function aliases](#) in the *AWS Lambda Developer Guide*.
10. Next, choose the **Configuration** tab and then **Permissions**.
11. Scroll down to **Resource-based policy** and then choose **Add permissions**. A resource-based policy is used to grant permissions to invoke your function to the principal that is specified in the policy. In this case, the principal will be the [Amazon EC2 Auto Scaling service-linked role](#) that is associated with the Auto Scaling group.
12. In the **Policy statement** section, configure your permissions:
 - a. Choose **AWS account**.
 - b. For **Principal**, enter the ARN of the calling service-linked role, for example, `arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling`.
 - c. For **Action**, choose **lambda:InvokeFunction**.
 - d. For **Statement ID**, enter a unique statement ID, such as **AllowInvokeByAutoScaling**.
 - e. Choose **Save**.
13. After you have followed these instructions, continue on to specify the ARN of your function in the termination policies for your Auto Scaling group as a next step. For more information, see [Use different termination policies \(console\) \(p. 235\)](#).

Note

For examples that you can use as a reference for developing your Lambda function, see the [GitHub repository for Amazon EC2 Auto Scaling](#).

Limitations

- You can only specify one Lambda function in the termination policies for an Auto Scaling group. If there are multiple termination policies specified, the Lambda function must be specified first.
- You can reference your Lambda function using either an unqualified ARN (without a suffix) or a qualified ARN that has either a version or an alias as its suffix. If an unqualified ARN is used (for example, `function:my-function`), your resource-based policy must be created on the unpublished version of your function. If a qualified ARN is used (for example, `function:my-function:1` or `function:my-function:prod`), your resource-based policy must be created on that specific published version of your function.
- You cannot use a qualified ARN with the `$LATEST` suffix. If you try to add a custom termination policy that refers to a qualified ARN with the `$LATEST` suffix, it will result in an error.
- The number of instances provided in the input data is limited to 30,000 instances. If there are more than 30,000 instances that could be terminated, the input data includes `"HasMoreInstances": true` to indicate that the maximum number of instances are returned.
- The maximum run time for your Lambda function is two seconds (2000 milliseconds). As a best practice, you should set the timeout value of your Lambda function based on your expected run time. Lambda functions have a default timeout of three seconds, but this can be decreased.
- Amazon EC2 Auto Scaling won't terminate instances that have instance scale-in protection enabled.

Use instance scale-in protection

To control whether an Auto Scaling group can terminate a particular instance when scaling in, use instance scale-in protection. You can enable the instance scale-in protection setting on an Auto Scaling group or on an individual Auto Scaling instance. When Amazon EC2 Auto Scaling launches a new instance or moves an instance from a warm pool into the Auto Scaling group, the instance inherits the instance scale-in protection setting of the Auto Scaling group. You can change the instance scale-in protection setting for an Auto Scaling group or an Auto Scaling instance at any time.

Instance scale-in protection starts when the instance state is `InService`. If you detach an instance that is protected from scale-in, its instance scale-in protection setting is lost. When you attach the instance to the group again, it inherits the current instance scale-in protection setting of the group.

Instance scale-in protection does not protect Auto Scaling instances from the following:

- Manual termination through the Amazon EC2 console, the `terminate-instances` command, or the `TerminateInstances` action. To protect Auto Scaling instances from manual termination, enable Amazon EC2 termination protection. For more information, see [Enabling termination protection](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Health check replacement if the instance fails health checks. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).
- Spot Instance interruptions. A Spot Instance is terminated when capacity is no longer available or the Spot price exceeds your maximum price.

Tasks

- [Enable instance scale-in protection for a group \(p. 241\)](#)
- [Modify the instance scale-in protection setting for a group \(p. 241\)](#)
- [Modify the instance scale-in protection setting for an instance \(p. 242\)](#)

Note

If all instances in an Auto Scaling group are protected from scale in, and a scale-in event occurs, its desired capacity is decremented. However, the Auto Scaling group can't terminate the required number of instances until their instance scale-in protection settings are disabled.

In the AWS Management Console, the **Activity history** for the Auto Scaling group includes the following message if all instances in an Auto Scaling group are protected from scale in when a scale-in event occurs: Could not scale to desired capacity because all remaining instances are protected from scale-in.

Enable instance scale-in protection for a group

You can enable instance scale-in protection when you create an Auto Scaling group. By default, instance scale-in protection is disabled.

To enable instance scale-in protection (console)

When you create the Auto Scaling group, on the **Configure group size and scaling policies** page, under **Instance scale-in protection**, select the **Enable instance scale-in protection** option.

To enable instance scale-in protection (AWS CLI)

Use the following [create-auto-scaling-group](#) command to enable instance scale-in protection.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in ...
```

Modify the instance scale-in protection setting for a group

You can enable or disable the instance scale-in protection setting for an Auto Scaling group. When the instance scale-in protection setting is enabled, all new instances launched after enabling it will have instance scale-in protection enabled. Previously launched instances are only protected from scale in if you enable the instance scale-in protection setting for each instance individually.

To change the instance scale-in protection setting for a group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select check box next to the Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Advanced configurations**, **Edit**.
4. For **Instance scale-in protection**, select **Enable instance scale-in protection**.
5. Choose **Update**.

To change the instance scale-in protection setting for a group (AWS CLI)

Use the following [update-auto-scaling-group](#) command to enable instance scale-in protection for the specified Auto Scaling group.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-instances-protected-from-scale-in
```

Use the following command to disable instance scale-in protection for the specified group.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-instances-protected-from-scale-in
```

Modify the instance scale-in protection setting for an instance

By default, an instance gets its instance scale-in protection setting from its Auto Scaling group. However, you can enable or disable instance scale-in protection for an instance at any time.

To change the instance scale-in protection setting for an instance (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Instance management** tab, in **Instances**, select an instance.
4. To enable instance scale-in protection, choose **Actions, Set scale-in protection**. When prompted, choose **Set scale-in protection**.
5. To disable instance scale-in protection, choose **Actions, Remove scale-in protection**. When prompted, choose **Remove scale-in protection**.

To change the instance scale-in protection setting for an instance (AWS CLI)

Use the following `set-instance-protection` command to enable instance scale-in protection for the specified instance.

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --protected-from-scale-in
```

Use the following command to disable instance scale-in protection for the specified instance.

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --no-protected-from-scale-in
```

Temporarily remove instances from your Auto Scaling group

You can put an instance that is in the `InService` state into the `Standby` state, update or troubleshoot the instance, and then return the instance to service. Instances that are on `standby` are still part of the Auto Scaling group, but they do not actively handle load balancer traffic.

This feature helps you stop and start the instances or reboot them without worrying about Amazon EC2 Auto Scaling terminating the instances as part of its health checks or during scale-in events.

For example, you can change the Amazon Machine Image (AMI) for an Auto Scaling group at any time by changing the launch template or launch configuration. Any subsequent instances that the Auto Scaling group launches use this AMI. However, the Auto Scaling group does not update the instances that are currently in service. You can terminate these instances and let Amazon EC2 Auto Scaling replace them, or use the instance refresh feature to terminate and replace the instances. Or, you can put the instances on `standby`, update the software, and then put the instances back in service.

Detaching instances from an Auto Scaling group is similar to putting instances on `standby`. Detaching instances might be useful if you want to manage the instances like standalone EC2 instances and possibly terminate them. For more information, see [Detach EC2 instances from your Auto Scaling group \(p. 130\)](#).

Important

When you put instances on standby, your Auto Scaling group can become unbalanced between Availability Zones. Amazon EC2 Auto Scaling compensates by rebalancing the Availability Zones unless you suspend the AZRebalance process. For more information, see [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#).

Contents

- [How the standby state works \(p. 243\)](#)
- [Health status of an instance in a standby state \(p. 244\)](#)
- [Temporarily remove an instance \(console\) \(p. 244\)](#)
- [Temporarily remove an instance \(AWS CLI\) \(p. 244\)](#)

Important

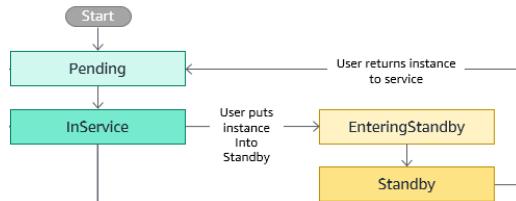
You are billed for instances that are in a standby state.

How the standby state works

The standby state works as follows to help you temporarily remove an instance from your Auto Scaling group:

1. You put the instance into the standby state. The instance remains in this state until you exit the standby state.
2. If there is a load balancer target group or Classic Load Balancer attached to your Auto Scaling group, the instance is deregistered from the load balancer. If connection draining is enabled for the load balancer, Elastic Load Balancing waits 300 seconds by default before completing the deregistration process, which helps in-flight requests to complete.
3. By default, the value that you specified as your desired capacity is decremented when you put an instance on standby. This prevents the launch of an additional instance while you have this instance on standby. Alternatively, you can specify that your desired capacity is not decremented. If you specify this option, the Auto Scaling group launches an instance to replace the one on standby. The intention is to help you maintain capacity for your application while one or more instances are on standby.
4. You can update or troubleshoot the instance.
5. You return the instance to service by exiting the standby state.
6. After you put an instance that was on standby back in service, the desired capacity is incremented. If you did not decrement the capacity when you put the instance on standby, the Auto Scaling group detects that you have more instances than you need. It applies the termination policy in effect to reduce the size of the group. For more information, see [Control which Auto Scaling instances terminate during scale in \(p. 230\)](#).
7. If there is a load balancer target group or Classic Load Balancer attached to your Auto Scaling group, the instance is registered with the load balancer.

The following illustration shows the transitions between instance states in this process:



For more information about the complete lifecycle of instances in an Auto Scaling group, see [Amazon EC2 Auto Scaling instance lifecycle \(p. 7\)](#).

Health status of an instance in a standby state

Amazon EC2 Auto Scaling does not perform health checks on instances that are in a standby state. While the instance is in a standby state, its health status reflects the status that it had before you put it on standby. Amazon EC2 Auto Scaling does not perform a health check on the instance until you put it back in service.

For example, if you put a healthy instance on standby and then terminate it, Amazon EC2 Auto Scaling continues to report the instance as healthy. If you attempt to put a terminated instance that was on standby back in service, Amazon EC2 Auto Scaling performs a health check on the instance, determines that it is terminating and unhealthy, and launches a replacement instance. For an introduction to health checks, see [Health checks for Auto Scaling instances \(p. 253\)](#).

Temporarily remove an instance (console)

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.
 3. On the **Instance management** tab, in **Instances**, select an instance.
 4. Choose **Actions, Set to Standby**.
 5. In the **Set to Standby** dialog box, select the check box to launch a replacement instance. Leave it unchecked to decrement the desired capacity. Choose **Set to Standby**.
 6. You can update or troubleshoot your instance as needed. When you have finished, continue with the next step to return the instance to service.
 7. Select the instance, choose **Actions, Set to InService**. In the **Set to InService** dialog box, choose **Set to InService**.

Temporarily remove an instance (AWS CLI)

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance

1. Use the following `describe-auto-scaling-instances` command to identify the instance to update.

```
aws autoscaling describe-auto-scaling-instances
```

The following is an example response.

```

        "Version": "1",
        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
    },
    "InstanceId": "i-05b4f7d5be44822a6",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService"
},
...
]
}

```

- Move the instance into a Standby state using the following [enter-standby](#) command. The `--should-decrement-desired-capacity` option decreases the desired capacity so that the Auto Scaling group does not launch a replacement instance.

```
aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
--auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

The following is an example response.

```
{
    "Activities": [
        {
            "Description": "Moving EC2 instance to Standby: i-05b4f7d5be44822a6",
            "AutoScalingGroupName": "my-asg",
            "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
            "Details": "{\"Availability Zone\":\"us-west-2a\"}",
            "StartTime": "2014-12-15T21:31:26.150Z",
            "Progress": 50,
            "Cause": "At 2014-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was moved to
standby
in response to a user request, shrinking the capacity from 4 to 3.",
            "StatusCode": "InProgress"
        }
    ]
}
```

- (Optional) Verify that the instance is in Standby using the following [describe-auto-scaling-instances](#) command.

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-05b4f7d5be44822a6
```

The following is an example response. Notice that the status of the instance is now Standby.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby"
        },
        ...
    ]
}
```

```
    ]  
}
```

4. You can update or troubleshoot your instance as needed. When you have finished, continue with the next step to return the instance to service.
5. Put the instance back in service using the following [exit-standby](#) command.

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
    "Activities": [  
        {  
            "Description": "Moving EC2 instance out of Standby: i-05b4f7d5be44822a6",  
            "AutoScalingGroupName": "my-asg",  
            "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",  
            "Details": "{\"Availability Zone\":\"us-west-2a\"}",  
            "StartTime": "2014-12-15T21:46:14.678Z",  
            "Progress": 30,  
            "Cause": "At 2014-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was moved  
out of standby in  
            response to a user request, increasing the capacity from 3 to 4.",  
            "StatusCode": "PreInService"  
        }  
    ]  
}
```

6. (Optional) Verify that the instance is back in service using the following [describe-auto-scaling-instances](#) command.

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-05b4f7d5be44822a6
```

The following is an example response. Notice that the status of the instance is `InService`.

```
{  
    "AutoScalingInstances": [  
        {  
            "ProtectedFromScaleIn": false,  
            "AvailabilityZone": "us-west-2a",  
            "LaunchTemplate": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "1",  
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"  
            },  
            "InstanceId": "i-05b4f7d5be44822a6",  
            "AutoScalingGroupName": "my-asg",  
            "HealthStatus": "HEALTHY",  
            "LifecycleState": "InService"  
        },  
        ...  
    ]  
}
```

Suspend and resume a process for an Auto Scaling group

This topic explains how to suspend and then resume one or more of the processes for your Auto Scaling group. You might want to do this, for example, so that you can investigate a configuration issue that is causing the process to fail, or to prevent Amazon EC2 Auto Scaling from marking instances unhealthy and replacing them while you are making changes to your Auto Scaling group.

Contents

- [Types of processes \(p. 247\)](#)
- [Considerations \(p. 248\)](#)
- [Suspend and resume processes \(console\) \(p. 250\)](#)
- [Suspend and resume processes \(AWS CLI\) \(p. 250\)](#)

Note

In addition to suspensions that you initiate, Amazon EC2 Auto Scaling can also suspend processes for Auto Scaling groups that repeatedly fail to launch instances. This is known as an *administrative suspension*. An administrative suspension most commonly applies to Auto Scaling groups that have been trying to launch instances for over 24 hours but have not succeeded in launching any instances. You can resume processes that were suspended by Amazon EC2 Auto Scaling for administrative reasons.

Types of processes

The suspend-resume feature supports the following processes:

- **Launch**—Adds instances to the Auto Scaling group when the group scales out, or when Amazon EC2 Auto Scaling chooses to launch instances for other reasons, such as when it adds instances to a warm pool.
- **Terminate**—Removes instances from the Auto Scaling group when the group scales in, or when Amazon EC2 Auto Scaling chooses to terminate instances for other reasons, such as when an instance is terminated for exceeding its maximum lifetime duration or failing a health check.
- **AddToLoadBalancer**—Adds instances to the attached load balancer target group or Classic Load Balancer when they are launched. For more information, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#).
- **AlarmNotification**—Accepts notifications from CloudWatch alarms that are associated with dynamic scaling policies. For more information, see [Dynamic scaling for Amazon EC2 Auto Scaling \(p. 133\)](#).
- **AZRebalance**—Balances the number of EC2 instances in the group evenly across all of the specified Availability Zones when the group becomes unbalanced, for example, when a previously unavailable Availability Zone returns to a healthy state. For more information, see [Rebalancing activities \(p. 6\)](#).
- **HealthCheck**—Checks the health of the instances and marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing tells Amazon EC2 Auto Scaling that the instance is unhealthy. This process can override the health status of an instance that you set manually. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).
- **InstanceRefresh**—Terminates and replaces instances using the instance refresh feature. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).
- **ReplaceUnhealthy**—Terminates instances that are marked as unhealthy and then creates new instances to replace them. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).

- **ScheduledActions**—Performs the scheduled scaling actions that you create or that are created for you when you create an AWS Auto Scaling scaling plan and turn on predictive scaling. For more information, see [Scheduled scaling for Amazon EC2 Auto Scaling \(p. 190\)](#).

Considerations

Consider the following before suspending processes:

- You can suspend and resume individual processes or all processes.
- Suspending a process affects all instances in your Auto Scaling group. For example, you can suspend the `HealthCheck` and `ReplaceUnhealthy` processes to reboot instances without Amazon EC2 Auto Scaling terminating the instances based on its health checks. If you need Amazon EC2 Auto Scaling to perform health checks on remaining instances, then use the standby feature instead of the suspend-resume feature. For more information, see [Temporarily remove instances from your Auto Scaling group \(p. 242\)](#).
- Suspending `AlarmNotification` allows you to temporarily stop the group's target tracking, step, and simple scaling policies without deleting the scaling policies or their associated CloudWatch alarms. To temporarily stop individual scaling policies instead, see [Disable a scaling policy for an Auto Scaling group \(p. 163\)](#).
- If you suspend the `Launch` and `Terminate` processes, or `AZRebalance`, and then you make changes to your Auto Scaling group, for example, by detaching instances or changing the Availability Zones that are specified, your group can become unbalanced between Availability Zones. If that happens, after you resume the suspended processes, Amazon EC2 Auto Scaling gradually redistributes instances evenly between the Availability Zones.
- Suspending the `Terminate` process doesn't prevent the successful termination of instances using the force delete option with the `delete-auto-scaling-group` command.

Understand how suspending processes affects other processes

The following descriptions explain what happens when individual process types are suspended.

Scenario 1: Launch is suspended

- `AlarmNotification` is still active, but your Auto Scaling group can't initiate scale-out activities for alarms that are in breach.
- `ScheduledActions` is active, but your Auto Scaling group can't initiate scale-out activities for any scheduled actions that occur.
- `AZRebalance` stops rebalancing the group.
- `ReplaceUnhealthy` continues to terminate unhealthy instances, but does not launch replacements. When you resume the `Launch` process, Amazon EC2 Auto Scaling immediately replaces any instances that it terminated during the time that `Launch` was suspended.
- `InstanceRefresh` does not replace instances.

Scenario 2: Terminate is suspended

- `AlarmNotification` is still active, but your Auto Scaling group can't initiate scale-in activities for alarms that are in breach.
- `ScheduledActions` is active, but your Auto Scaling group can't initiate scale-in activities for any scheduled actions that occur.
- `AZRebalance` is still active but does not function properly. It can launch new instances without terminating the old ones. This could cause your Auto Scaling group to grow up to 10 percent larger

than its maximum size, because this is allowed temporarily during rebalancing activities. Your Auto Scaling group could remain above its maximum size until you resume the `Terminate` process.

- `ReplaceUnhealthy` is inactive but not `HealthCheck`. When `Terminate` resumes, the `ReplaceUnhealthy` process immediately starts running. If any instances were marked as unhealthy while `Terminate` was suspended, they are immediately replaced.
- `InstanceRefresh` does not replace instances.

Scenario 3: `AddToLoadBalancer` is suspended

- Amazon EC2 Auto Scaling launches the instances but does not add them to the load balancer target group or Classic Load Balancer. When you resume the `AddToLoadBalancer` process, it resumes adding instances to the load balancer when they are launched. However, it does not add the instances that were launched while this process was suspended. You must register those instances manually.

Scenario 4: `AlarmNotification` is suspended

- Amazon EC2 Auto Scaling does not invoke scaling policies when a CloudWatch alarm threshold is in breach. When you resume `AlarmNotification`, Amazon EC2 Auto Scaling considers policies with alarm thresholds that are currently in breach.

Scenario 5: `AZRebalance` is suspended

- Amazon EC2 Auto Scaling does not attempt to redistribute instances after certain events. However, if a scale-out or scale-in event occurs, the scaling process still tries to balance the Availability Zones. For example, during scale out, it launches the instance in the Availability Zone with the fewest instances. If the group becomes unbalanced while `AZRebalance` is suspended and you resume it, Amazon EC2 Auto Scaling attempts to rebalance the group. It first calls `Launch` and then `Terminate`.

Scenario 6: `HealthCheck` is suspended

- Amazon EC2 Auto Scaling stops marking instances unhealthy as a result of EC2 and Elastic Load Balancing health checks. Your custom health checks continue to function properly. After you suspend `HealthCheck`, if you need to, you can manually set the health state of instances in your group and have `ReplaceUnhealthy` replace them.

Scenario 7: `InstanceRefresh` is suspended

- Amazon EC2 Auto Scaling stops replacing instances as a result of an instance refresh. If there is an instance refresh in progress, this pauses the operation without canceling it.

Scenario 8: `ReplaceUnhealthy` is suspended

- Amazon EC2 Auto Scaling stops replacing instances that are marked as unhealthy. Instances that fail EC2 or Elastic Load Balancing health checks are still marked as unhealthy. As soon as you resume the `ReplaceUnhealthy` process, Amazon EC2 Auto Scaling replaces instances that were marked unhealthy while this process was suspended. The `ReplaceUnhealthy` process calls `Terminate` first and then `Launch`.

Scenario 9: `ScheduledActions` is suspended

- Amazon EC2 Auto Scaling does not run scheduled actions that are scheduled to run during the suspension period. When you resume `ScheduledActions`, Amazon EC2 Auto Scaling only considers scheduled actions whose scheduled time has not yet passed.

Additional considerations

In addition, when Launch or Terminate are suspended, the following features might not function correctly:

- **Maximum instance lifetime**—When Launch or Terminate are suspended, the maximum instance lifetime feature can't replace any instances.
- **Spot Instance interruptions**—If Terminate is suspended and your Auto Scaling group has Spot Instances, they can still terminate in the event that Spot capacity is no longer available. While Launch is suspended, Amazon EC2 Auto Scaling can't launch replacement instances from another Spot Instance pool or from the same Spot Instance pool when it is available again.
- **Capacity Rebalancing**—If Terminate is suspended and you use Capacity Rebalancing to handle Spot Instance interruptions, the Amazon EC2 Spot service can still terminate instances in the event that Spot capacity is no longer available. If Launch is suspended, Amazon EC2 Auto Scaling can't launch replacement instances from another Spot Instance pool or from the same Spot Instance pool when it is available again.
- **Attaching and detaching instances**—When Launch and Terminate are suspended, you can detach instances that are attached to your Auto Scaling group, but while Launch is suspended, you can't attach new instances to the group.
- **Standby instances**—When Launch and Terminate are suspended, you can put an instance in the Standby state, but while Launch is suspended, you can't return an instance in the Standby state to service.

Suspend and resume processes (console)

Use the following procedure to suspend a process.

To suspend a process

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. Select the check box next to the Auto Scaling group.
- A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Advanced configurations, Edit**.
 4. For **Suspended processes**, choose the process to suspend.
 5. Choose **Update**.

When you are ready, use the following procedure to resume the suspended process.

To resume a process

1. On the **Details** tab, choose **Advanced configurations, Edit**.
2. For **Suspended processes**, remove the suspended process.
3. Choose **Update**.

Suspend and resume processes (AWS CLI)

Use the following `suspend-processes` command to suspend individual processes.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-processes HealthCheck ReplaceUnhealthy
```

To suspend all processes, omit the `--scaling-processes` option, as follows.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

When you are ready to resume a suspended process, use the following [resume-processes](#) command.

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-  
processes HealthCheck
```

To resume all suspended processes, omit the `--scaling-processes` option, as follows.

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

Monitor your Auto Scaling instances and groups

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon EC2 Auto Scaling and your Amazon Web Services Cloud solutions. AWS provides the following monitoring tools to watch Amazon EC2 Auto Scaling, report when something is wrong, and take automatic actions when appropriate:

Health checks

Amazon EC2 Auto Scaling periodically performs health checks on the instances in your Auto Scaling group. If an instance does not pass its health check, it is marked unhealthy and will be terminated while Amazon EC2 Auto Scaling launches a new instance in replacement. For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).

AWS Health Dashboard

The AWS Health Dashboard displays information, and also provides notifications that are triggered by changes in the health of AWS resources. The information is presented in two ways: on a dashboard that shows recent and upcoming events organized by category, and in a full event log that shows all events from the past 90 days. For more information, see [AWS Health Dashboard notifications for Amazon EC2 Auto Scaling \(p. 259\)](#).

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 [Monitor CloudWatch metrics for your Auto Scaling groups and instances \(p. 260\)](#).

CloudWatch dashboards

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 the [Amazon CloudWatch User Guide](#).

CloudTrail logs

AWS CloudTrail enables you to track the calls made to the Amazon EC2 Auto Scaling API by or on behalf of your AWS account. CloudTrail stores the information in log files in the Amazon S3 bucket that you specify. You can use these log files to monitor activity of your Auto Scaling groups. Logs include which requests were made, the source IP addresses where the requests came from, who made the request, when the request was made, and so on. For more information, see [Log Amazon EC2 Auto Scaling API calls with AWS CloudTrail \(p. 269\)](#).

CloudWatch Logs

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](#).

Amazon Simple Notification Service notifications

You can configure Auto Scaling groups to send Amazon SNS notifications when Amazon EC2 Auto Scaling launches or terminates instances. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).

Health checks for Auto Scaling instances

The health status of an Auto Scaling instance indicates whether it is healthy or unhealthy. All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless Amazon EC2 Auto Scaling receives notification that they are unhealthy. This notification can come from sources such as Amazon EC2, Elastic Load Balancing, or custom health checks. When Amazon EC2 Auto Scaling detects an unhealthy instance, it terminates it and launches a new one.

Contents

- [Health check types \(p. 253\)](#)
- [Amazon EC2 health checks \(p. 253\)](#)
- [Elastic Load Balancing health checks \(p. 254\)](#)
- [Custom health detection tasks \(p. 255\)](#)
- [Unhealthy instance replacement \(p. 256\)](#)
- [How Amazon EC2 Auto Scaling minimizes downtime \(p. 256\)](#)
- [Health check considerations \(p. 257\)](#)
- [Additional information \(p. 257\)](#)
- [Set the health check grace period for an Auto Scaling group \(p. 258\)](#)

Health check types

Amazon EC2 Auto Scaling can determine the health status of an instance by using one or more of the following health checks:

Health check type	What it checks
Amazon EC2 status checks and scheduled events	<ul style="list-style-type: none">• Checks that the instance is running• Checks for underlying hardware or software issues that might impair the instance <p>This is the default health check type for an Auto Scaling group.</p>
Elastic Load Balancing health checks	<ul style="list-style-type: none">• Checks whether the load balancer reports the instance as healthy, confirming whether the instance is available to handle requests <p>To run this health check type, you must enable it for your Auto Scaling group.</p>
Custom health checks	<ul style="list-style-type: none">• Checks for any other problems that might indicate instance health issues, according to your custom health checks

Amazon EC2 health checks

After an instance launches, it is attached to the Auto Scaling group and enters the `InService` state. For more information about the different lifecycle states for instances in an Auto Scaling group, see [Amazon EC2 Auto Scaling instance lifecycle \(p. 7\)](#).

Amazon EC2 Auto Scaling periodically checks the health status of all instances within the Auto Scaling group to make sure that they're running and in good condition.

Status checks

Amazon EC2 Auto Scaling uses the results of the Amazon EC2 instance status checks and system status checks to determine the health status of an instance. If the instance is in any Amazon EC2 state other than `running`, or if its status for the status checks becomes `impaired`, Amazon EC2 Auto Scaling considers the instance to be unhealthy and replaces it. This includes when the instance has any of the following states:

- `stopping`
- `stopped`
- `shutting-down`
- `terminated`

The Amazon EC2 status checks do not require any special configuration and are always enabled. For more information, see [Types of status checks](#) in the *Amazon EC2 User Guide for Linux Instances*.

Important

Amazon EC2 Auto Scaling lets the status checks fail occasionally, without taking any action. When a status check fails, Amazon EC2 Auto Scaling waits a few minutes for AWS to fix the issue. It does not immediately mark an instance as unhealthy when its status for the status checks becomes `impaired`.

However, if Amazon EC2 Auto Scaling detects that an instance is no longer in the `running` state, this situation is treated as an immediate failure. In this case, it immediately marks the instance as unhealthy and replaces it.

Scheduled events

Amazon EC2 can occasionally schedule events on your instances to be run after a particular timestamp. For more information, see [Scheduled events for your instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

If one of your instances is affected by a scheduled event, Amazon EC2 Auto Scaling considers the instance to be unhealthy and replaces it. The instance doesn't start terminating until the date and time specified in the timestamp is reached.

Elastic Load Balancing health checks

When you enable Elastic Load Balancing health checks for your Auto Scaling group, Amazon EC2 Auto Scaling can use the results of those health checks to determine the health status of an instance.

Before you can enable Elastic Load Balancing health checks for your Auto Scaling group, you must do the following:

- Set up an Elastic Load Balancing load balancer and configure a health check for it to use to determine if your instances are healthy.
- Attach the load balancer to your Auto Scaling group.

After you complete the preceding actions, the following occurs:

- Amazon EC2 Auto Scaling registers the instances in the Auto Scaling group with the load balancer.
- After an instance finishes registering, it enters the `InService` state and becomes available for use with the load balancer.

By default, Amazon EC2 Auto Scaling ignores the results of the Elastic Load Balancing health checks. However, you can enable these health checks for your Auto Scaling group. After you do this, when

Elastic Load Balancing reports a registered instance as unhealthy, Amazon EC2 Auto Scaling marks the instance as unhealthy on its next periodic health check and replaces it.

If connection draining (deregistration delay) is enabled for your load balancer, Amazon EC2 Auto Scaling waits for either in-flight requests to complete or the maximum timeout to expire before it terminates unhealthy instances.

For more information about how to enable Elastic Load Balancing health checks for your Auto Scaling group, see [Add Elastic Load Balancing health checks to an Auto Scaling group \(p. 291\)](#).

Note

When you enable Elastic Load Balancing health checks for a group, Amazon EC2 Auto Scaling can terminate and replace instances that are reported as unhealthy, but only after the load balancer is in the `InService` state. For more information, see [Understand the attachment status of your load balancer \(p. 290\)](#).

Custom health detection tasks

You might also want to run custom health detection tasks on the instances in your Auto Scaling group and set the health status of an instance as unhealthy if the task fails. This extends your health checks by using a combination of custom health checks, Amazon EC2 status checks, and Elastic Load Balancing health checks, if enabled.

You can send the instance health information directly to Amazon EC2 Auto Scaling using the AWS CLI or an SDK. The following examples show how to use the AWS CLI to configure the health state of an instance and then verify the instance's health state.

Use the following `set-instance-health` command to set the health state of the specified instance to `Unhealthy`.

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-status Unhealthy
```

By default, this command respects the health check grace period. However, you can override this behavior and not respect the grace period by including the `--no-should-respect-grace-period` option.

Use the following `describe-auto-scaling-groups` command to verify that the instance state is `Unhealthy`.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

The following is an example response that shows that the health status of the instance is `Unhealthy` and that the instance is terminating.

```
{  
    "AutoScalingGroups": [  
        {  
            ....  
            "Instances": [  
                {  
                    "ProtectedFromScaleIn": false,  
                    "AvailabilityZone": "us-west-2a",  
                    "LaunchTemplate": {  
                        "LaunchTemplateName": "my-launch-template",  
                        "Version": "1",  
                        "LaunchTemplateId": "lt-1234567890abcdef0"  
                    },  
                },  
            ]  
        }  
    ]  
}
```

```
        "InstanceId": "i-1234567890abcdef0",
        "HealthStatus": "Unhealthy",
        "LifecycleState": "Terminating"
    },
    ...
]
}
}
```

Unhealthy instance replacement

When Amazon EC2 Auto Scaling determines that an **InService** instance is unhealthy, it terminates the instance while it launches a new replacement instance. The new instance launches using the current settings of the Auto Scaling group and its associated launch template or launch configuration.

Amazon EC2 Auto Scaling creates a new scaling activity for terminating the unhealthy instance and then terminates it. While the instance is terminating, another scaling activity launches a new instance.

To view the reason for health check failures (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.

3. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances.

If it terminated any unhealthy instances, the **Cause** column shows the date and time of the termination and the reason for the health check failure. For example, At 2022-05-14T20:11:53Z an instance was taken out of service in response to an ELB system health check failure.

How Amazon EC2 Auto Scaling minimizes downtime

Health check replacements require instances to terminate first, which might prevent new requests from being accepted until new instances launch.

If Amazon EC2 Auto Scaling determines that any instances are no longer running or were marked unhealthy with the [set-instance-health](#) command, it immediately replaces them. However, if other instances are found to be unhealthy, Amazon EC2 Auto Scaling uses the following approach to recover from failures. This approach minimizes any downtime that might occur because of temporary issues or misconfigured health checks.

- If a scaling activity is in progress and your Auto Scaling group is below its desired capacity by 10 percent or more, Amazon EC2 Auto Scaling waits for the in-progress scaling activity before replacing the unhealthy instances.
- When scaling out, Amazon EC2 Auto Scaling waits for the instances to pass an initial health check. It also waits for the default instance warmup to finish to make sure that the new instances are ready.
- After the instances finish warming up and the group has risen to above 90 percent of its desired capacity, Amazon EC2 Auto Scaling replaces the unhealthy instances as follows:
 - Amazon EC2 Auto Scaling only replaces up to 10 percent of the group's desired capacity at a time. It does this until all of the unhealthy instances are replaced.

- When replacing instances, it waits for the new instances to pass an initial health check. It also waits for the default instance warmup to finish before continuing.

Note

If the size of an Auto Scaling group is small enough that the resulting value of 10 percent is less than one, Amazon EC2 Auto Scaling replaces the unhealthy instances one at a time instead. This might result in some downtime for the group.

Also, if all instances in an Auto Scaling group are reported as unhealthy by Elastic Load Balancing health checks and the load balancer is in the `InService` state, Amazon EC2 Auto Scaling might mark fewer instances unhealthy at a time. This can result in much fewer instances replaced at a time than the 10 percent applied in other scenarios. This provides you time to address the problem without Amazon EC2 Auto Scaling automatically terminating the entire group.

Health check considerations

This section contains considerations for Amazon EC2 Auto Scaling health checks.

- If you need something to happen on the instance that is terminating, or on the instance that is starting up, you can use lifecycle hooks. These hooks let you perform a custom action as Amazon EC2 Auto Scaling launches or terminates instances. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).
- Amazon EC2 Auto Scaling does not provide a way of removing the Amazon EC2 status checks and scheduled events from its health checks. If you do not want instances to be replaced, we recommend that you suspend the `ReplaceUnhealthy` and `HealthCheck` process for individual Auto Scaling groups. For more information, see [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#).
- To manually set an unhealthy instance's health status back to healthy, you can try to use the `set-instance-health` command. If you get an error, this is probably because the instance is already terminating. Generally, setting an instance's health status back to healthy with the `set-instance-health` command is only useful in cases where either the `ReplaceUnhealthy` process or the `Terminate` process is suspended.
- Amazon EC2 Auto Scaling does not perform health checks on instances that are in the `Standby` state. For more information, see [Temporarily remove instances from your Auto Scaling group \(p. 242\)](#).
- When your instance is terminated, any associated Elastic IP addresses are disassociated and are not automatically associated with the new instance. You must manually associate the Elastic IP addresses with the new instance, or do it automatically with a lifecycle hook-based solution. For more information, see [Elastic IP addresses](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Similarly, when your instance is terminated, its attached EBS volumes are detached (or deleted depending on the volume's `DeleteOnTermination` attribute). You must manually attach these EBS volumes to the new instance, or do it automatically with a lifecycle hook-based solution. For more information, see [Attach an Amazon EBS volume to an instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Additional information

For information about troubleshooting health checks, see [Troubleshoot Amazon EC2 Auto Scaling: Health checks \(p. 372\)](#). If your health checks fail, check this topic for troubleshooting steps. This topic will help you figure out what has gone wrong in your Auto Scaling group and give you suggestions on how to fix it.

Amazon EC2 Auto Scaling also monitors the health of instances that you launch into a warm pool using Amazon EC2, Amazon EBS, or custom health checks. For more information, see [View health check status and the reason for health check failures \(p. 226\)](#).

Set the health check grace period for an Auto Scaling group

The health check grace period specifies the minimum amount of time (in seconds) to keep a new instance in service before terminating if it's unhealthy. A specific use case might be a requirement to wait to act on Elastic Load Balancing health checks because the instance is still initializing. The grace period prevents Amazon EC2 Auto Scaling from marking your newly launched instances unhealthy and terminating them unnecessarily if they do not immediately pass these health checks after they enter the `InService` state.

In the console, by default, the health check grace period is 300 seconds when you create an Auto Scaling group. Its default value is 0 seconds when you create an Auto Scaling group using the AWS CLI or an SDK.

Setting this value too high reduces the effectiveness of the Amazon EC2 Auto Scaling health checks. If you use lifecycle hooks for instance launch, you can set the health check grace period to 0. With lifecycle hooks, Amazon EC2 Auto Scaling provides a way to make sure that instances are always initialized before they enter the `InService` state. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

The grace period applies to the following instances:

- Newly launched instances
- Instances that are put back into service after being in standby
- Instances that you manually attach to the group

Important

During the health check grace period, if Amazon EC2 Auto Scaling detects that an instance is no longer in the Amazon EC2 running state, it immediately marks the instance as unhealthy and replaces it. For example, if you stop an instance in an Auto Scaling group, it is marked unhealthy and replaced.

Set the health check grace period for a group

You can set the health check grace period for new and existing Auto Scaling groups.

Console

To modify the health check grace period for a new group (console)

When you create the Auto Scaling group, on the **Configure advanced options** page, for **Health checks, Health check grace period**, enter the amount of time, in seconds. This is how long Amazon EC2 Auto Scaling needs to wait before checking the health status of an instance after it enters the `InService` state.

AWS CLI

To modify the health check grace period for a new group (AWS CLI)

Add the `--health-check-grace-period` option to the [create-auto-scaling-group](#) command. The following example configures the health check grace period with a value of `60` seconds for a new Auto Scaling group named `my-asg`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --health-check-grace-period 60 ...
```

Console

To modify the health check grace period for an existing group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. On the navigation bar at the top of the screen, choose the AWS Region that you created your Auto Scaling group in.
 3. Select the check box next to the Auto Scaling group.
- A split pane opens up in the bottom of the **Auto Scaling groups** page.
4. On the **Details** tab, choose **Health checks, Edit**.
 5. Under **Health check grace period**, enter the amount of time, in seconds. This is how long Amazon EC2 Auto Scaling needs to wait before checking the health status of an instance after it enters the **InService** state.
 6. Choose **Update**.

AWS CLI

To modify the health check grace period for an existing group (AWS CLI)

Add the `--health-check-grace-period` option to the `update-auto-scaling-group` command. The following example configures the health check grace period with a value of `120` seconds for an existing Auto Scaling group named `my-asg`.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --health-check-grace-period 120
```

Note

We strongly recommend also setting the default instance warm-up time for your Auto Scaling group. For a consolidated view of all your instance warm-up, health check grace period, and cooldown settings in one place, see [Available warm-up and cooldown settings \(p. 155\)](#).

AWS Health Dashboard notifications for Amazon EC2 Auto Scaling

Your AWS Health Dashboard provides support for notifications that come from Amazon EC2 Auto Scaling. These notifications provide awareness and remediation guidance for resource performance or availability issues that may affect your applications. Only events that are specific to missing security groups and launch templates are currently available.

The AWS Health Dashboard is part of the AWS Health service. It requires no set up and can be viewed by any user that is authenticated in your account. For more information, see [Getting started with the AWS Health Dashboard](#).

If you receive a message similar to the following messages, it should be treated as an alarm to take action.

Example: Auto Scaling group is not scaling out due to a missing security group

```
Hello,
```

At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS account 123456789012.

A security group associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration to change the launch template or launch configuration that depends on the unavailable security group.

Sincerely,
Amazon Web Services

Example: Auto Scaling group is not scaling out due to a missing launch template

Hello,

At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in AWS account 123456789012.

The launch template associated with this Auto Scaling group cannot be found. Each time a scale out operation is performed, it will be prevented until you make a change that fixes the issue.

We recommend that you review and update your Auto Scaling group configuration and specify an existing launch template to use.

Sincerely,
Amazon Web Services

Monitor CloudWatch metrics for your Auto Scaling groups and instances

Metrics are the fundamental concept in Amazon CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor, and the data points as representing the values of that variable over time. You can use these metrics to verify that your system is performing as expected.

Amazon EC2 Auto Scaling metrics that collect information about Auto Scaling groups are in the AWS/AutoScaling namespace. Amazon EC2 instance metrics that collect CPU and other usage data from Auto Scaling instances are in the AWS/EC2 namespace.

The Amazon EC2 Auto Scaling console displays a series of graphs for the group metrics and the aggregated instance metrics for the group. Depending on your needs, you might prefer to access data for your Auto Scaling groups and instances from Amazon CloudWatch instead of the Amazon EC2 Auto Scaling console.

For more information, see the [Amazon CloudWatch User Guide](#).

Contents

- [Available metrics and dimensions \(p. 261\)](#)
- [Enable Auto Scaling group metrics \(console\) \(p. 264\)](#)
- [Enable Auto Scaling group metrics \(AWS CLI\) \(p. 265\)](#)
- [Configure monitoring for Auto Scaling instances \(p. 265\)](#)

- [View monitoring graphs in the Amazon EC2 Auto Scaling console \(p. 267\)](#)

Available metrics and dimensions

Note

This section lists the different types of metrics in the AWS/AutoScaling namespace. For information about the available metrics in the AWS/EC2 namespace, see [List the available CloudWatch metrics for your instances in the Amazon EC2 User Guide for Linux Instances](#). See [Configure monitoring for Auto Scaling instances \(p. 265\)](#) to learn how to enable detailed monitoring for metrics in the AWS/EC2 namespace or collect memory metrics from the EC2 instances in your Auto Scaling groups.

Amazon EC2 Auto Scaling publishes the following metrics in the AWS/AutoScaling namespace.

Amazon EC2 Auto Scaling sends sampled data to CloudWatch every minute on a best-effort basis. In rare cases when CloudWatch experiences a service disruption, data isn't backfilled to fill gaps in group metric history.

Contents

- [Auto Scaling group metrics \(p. 261\)](#)
- [Dimensions for Auto Scaling group metrics \(p. 263\)](#)
- [Predictive scaling metrics and dimensions \(p. 264\)](#)

Auto Scaling group metrics

When group metrics are enabled, Amazon EC2 Auto Scaling sends the following metrics to CloudWatch. The metrics are available at one-minute granularity at no additional charge, but you must enable them. With these metrics, you get nearly continuous visibility into the history of your Auto Scaling group, such as changes in the size of the group over time.

Metric	Description
GroupMinSize	The minimum size of the Auto Scaling group. Reporting criteria: Reported if metrics collection is enabled.
GroupMaxSize	The maximum size of the Auto Scaling group. Reporting criteria: Reported if metrics collection is enabled.
GroupDesiredCapacity	The number of instances that the Auto Scaling group attempts to maintain. Reporting criteria: Reported if metrics collection is enabled.
GroupInServiceInstances	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating. Reporting criteria: Reported if metrics collection is enabled.
GroupPendingInstances	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating. Reporting criteria: Reported if metrics collection is enabled.

Metric	Description
GroupStandbyInstances	The number of instances that are in a Standby state. Instances in this state are still running but are not actively in service. Reporting criteria: Reported if metrics collection is enabled.
GroupTerminatingInstances	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending. Reporting criteria: Reported if metrics collection is enabled.
GroupTotalInstances	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating. Reporting criteria: Reported if metrics collection is enabled.

In addition to the metrics in the previous table, Amazon EC2 Auto Scaling also reports group metrics as an aggregate count of the number of capacity units that each instance represents. If instance weighting is not applied, then the following metrics are populated, but are equal to the metrics that are defined in the previous table. For more information about using weights, see [Configure instance weighting for Amazon EC2 Auto Scaling \(p. 68\)](#) and [Create an Auto Scaling group using attribute-based instance type selection \(p. 75\)](#).

Metric	Description
GroupInServiceCapacity	The number of capacity units that are running as part of the Auto Scaling group. Reporting criteria: Reported if metrics collection is enabled.
GroupPendingCapacity	The number of capacity units that are pending. Reporting criteria: Reported if metrics collection is enabled.
GroupStandbyCapacity	The number of capacity units that are in a Standby state. Reporting criteria: Reported if metrics collection is enabled.
GroupTerminatingCapacity	The number of capacity units that are in the process of terminating. Reporting criteria: Reported if metrics collection is enabled.
GroupTotalCapacity	The total number of capacity units in the Auto Scaling group. Reporting criteria: Reported if metrics collection is enabled.

Amazon EC2 Auto Scaling also reports the following metrics for Auto Scaling groups that have a warm pool. For more information, see [Warm pools for Amazon EC2 Auto Scaling \(p. 219\)](#).

Metric	Description
WarmPoolMinSize	The minimum size of the warm pool. Reporting criteria: Reported if metrics collection is enabled.

Metric	Description
WarmPoolDesiredCapacity	<p>The amount of capacity that Amazon EC2 Auto Scaling attempts to maintain in the warm pool.</p> <p>This is equivalent to the maximum size of the Auto Scaling group minus its desired capacity, or, if set, as the maximum prepared capacity of the Auto Scaling group minus its desired capacity.</p> <p>However, when the minimum size of the warm pool is equal to or greater than the difference between the maximum size (or, if set, the maximum prepared capacity) and the desired capacity of the Auto Scaling group, then the warm pool desired capacity will be equivalent to the WarmPoolMinSize.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
WarmPoolPendingCapacity	<p>The amount of capacity in the warm pool that is pending. This metric does not include instances that are running, stopped, or terminating.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
WarmPoolTerminatingCapacity	<p>The amount of capacity in the warm pool that is in the process of terminating. This metric does not include instances that are running, stopped, or pending.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
WarmPoolWarmedCapacity	<p>The amount of capacity available to enter the Auto Scaling group during scale out. This metric does not include instances that are pending or terminating.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
WarmPoolTotalCapacity	<p>The total capacity of the warm pool, including instances that are running, stopped, pending, or terminating.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
GroupAndWarmPoolDesiredCapacity	<p>The desired capacity of the Auto Scaling group and the warm pool combined.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>
GroupAndWarmPoolTotalCapacity	<p>The total capacity of the Auto Scaling group and the warm pool combined. This includes instances that are running, stopped, pending, terminating, or in service.</p> <p>Reporting criteria: Reported if metrics collection is enabled.</p>

Dimensions for Auto Scaling group metrics

You can use the following dimensions to refine the metrics listed in the previous tables.

Dimension	Description
AutoScalingGroupName	Filters on the name of an Auto Scaling group.

Predictive scaling metrics and dimensions

The AWS/AutoScaling namespace includes the following metrics for predictive scaling.

Metrics are available with a resolution of one hour.

You can evaluate forecast accuracy by comparing forecasted values with actual values. For more information about evaluating forecast accuracy using these metrics, see [Monitor predictive scaling metrics with CloudWatch \(p. 177\)](#).

Metric	Description	Dimensions
PredictiveScalingLoad	<p>The amount of load that's anticipated to be generated by your application.</p> <p>The Average, Minimum, and Maximum statistics are useful, but the Sum statistic is not.</p> <p>Reporting criteria: Reported after the initial forecast is created.</p>	AutoScalingGroupName, PolicyName, PairIndex
PredictiveScalingCapacity	<p>The anticipated amount of capacity needed to meet application demand. This is based on the load forecast and target utilization level at which you want to maintain your Auto Scaling instances.</p> <p>The Average, Minimum, and Maximum statistics are useful, but the Sum statistic is not.</p> <p>Reporting criteria: Reported after the initial forecast is created.</p>	AutoScalingGroupName, PolicyName

Note

The `PairIndex` dimension returns information associated with the index of the load-scaling metric pair as assigned by Amazon EC2 Auto Scaling. Currently, the only valid value is `0`.

Enable Auto Scaling group metrics (console)

When you enable Auto Scaling group metrics, your Auto Scaling group sends sampled data to CloudWatch every minute. There is no charge for enabling these metrics.

To enable group metrics

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom of the page.
3. On the **Monitoring** tab, select the **Auto Scaling group metrics collection**, **Enable** check box located at the top of the page under **Auto Scaling**.

To disable group metrics

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.

2. Select your Auto Scaling group.
3. On the **Monitoring** tab, clear the **Auto Scaling group metrics collection, Enable** check box.

Enable Auto Scaling group metrics (AWS CLI)

To enable group metrics

Enable one or more group metrics by using the [enable-metrics-collection](#) command. For example, the following command enables the `GroupDesiredCapacity` metric.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--metrics GroupDesiredCapacity --granularity "1Minute"
```

If you omit the `--metrics` option, all metrics are enabled.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \  
--granularity "1Minute"
```

To disable group metrics

Use the [disable-metrics-collection](#) command. For example, the following command disables all Auto Scaling group metrics.

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

Configure monitoring for Auto Scaling instances

Amazon EC2 collects and processes raw data from instances into readable, near real-time metrics that describe the CPU and other usage data for your Auto Scaling group. You can configure the interval for monitoring these metrics by choosing one-minute or five-minute granularity.

Instance monitoring is enabled whenever an instance is launched, using either basic monitoring (five-minute granularity) or detailed monitoring (one-minute granularity). For detailed monitoring, additional charges apply. For more information, see [Amazon CloudWatch pricing](#) and [Monitoring your instances using CloudWatch](#) in the *Amazon EC2 User Guide for Linux Instances*.

Before creating an Auto Scaling group, you should create a launch template or launch configuration that permits the type of monitoring that is appropriate to your application. If you add a scaling policy to your group, we strongly recommend that you use detailed monitoring to get metric data for EC2 instances at a one-minute granularity, because that achieves a faster response to changes in load.

Contents

- [Enable detailed monitoring \(console\) \(p. 265\)](#)
- [Enable detailed monitoring \(AWS CLI\) \(p. 266\)](#)
- [Switch between basic and detailed monitoring \(p. 266\)](#)
- [Collect additional metrics using the CloudWatch agent \(p. 266\)](#)

Enable detailed monitoring (console)

By default, basic monitoring is enabled when you use the AWS Management Console to create a launch template or launch configuration.

To enable detailed monitoring in a launch template

When you create the launch template using the AWS Management Console, in the **Advanced details** section, for **Detailed CloudWatch monitoring**, choose **Enable**. Otherwise, basic monitoring is enabled. For more information, see [Configure advanced settings for your launch template \(p. 27\)](#).

To enable detailed monitoring in a launch configuration

When you create the launch configuration using the AWS Management Console, in the **Additional configuration** section, select **Enable EC2 instance detailed monitoring within CloudWatch**. Otherwise, basic monitoring is enabled. For more information, see [Create a launch configuration \(p. 40\)](#).

Enable detailed monitoring (AWS CLI)

By default, basic monitoring is enabled when you create a launch template using the AWS CLI. Detailed monitoring is enabled by default when you create a launch configuration using the AWS CLI.

To enable detailed monitoring in a launch template

For launch templates, use the [create-launch-template](#) command and pass a JSON file that contains the information for creating the launch template. Set the monitoring attribute to "Monitoring": {"Enabled": true} to enable detailed monitoring or "Monitoring": {"Enabled": false} to enable basic monitoring.

To enable detailed monitoring in a launch configuration

For launch configurations, use the [create-launch-configuration](#) command with the --instance-monitoring option. Set this option to true to enable detailed monitoring or false to enable basic monitoring.

```
--instance-monitoring Enabled=true
```

Switch between basic and detailed monitoring

To change the type of monitoring enabled on new EC2 instances, update the launch template or update the Auto Scaling group to use a new launch template or launch configuration. Existing instances continue to use the previously enabled monitoring type. To update all instances, terminate them so that they are replaced by your Auto Scaling group or update instances individually using [monitor-instances](#) and [unmonitor-instances](#).

Note

With the instance refresh and maximum instance lifetime features, you can also replace all instances in the Auto Scaling group to launch new instances that use the new settings. For more information, see [Replace Auto Scaling instances \(p. 99\)](#).

When you switch between basic and detailed monitoring:

If you have CloudWatch alarms associated with your Auto Scaling group, use the [put-metric-alarm](#) command to update each alarm. Make each period match the monitoring type (300 seconds for basic monitoring and 60 seconds for detailed monitoring). If you change from detailed monitoring to basic monitoring but do not update your alarms to match the five-minute period, they continue to check for statistics every minute. They might find no data available for as many as four out of every five periods.

Collect additional metrics using the CloudWatch agent

To collect operating system-level metrics like available and used memory, you must install the CloudWatch agent. Additional fees may apply. For more information, see [Metrics collected by the CloudWatch agent](#) in the *Amazon CloudWatch User Guide*.

View monitoring graphs in the Amazon EC2 Auto Scaling console

In the Amazon EC2 Auto Scaling section of the Amazon EC2 console, you can monitor minute-by-minute progress of individual Auto Scaling groups using CloudWatch metrics.

You can monitor the following types of metrics:

- **Auto Scaling metrics** – Auto Scaling metrics are turned on only when you enable them. For more information, see [Enable Auto Scaling group metrics \(console\) \(p. 264\)](#). When Auto Scaling metrics are enabled, the monitoring graphs show data published at one-minute granularity for Auto Scaling metrics.
- **EC2 metrics** – If detailed monitoring is enabled, the monitoring graphs show data published at one-minute granularity for instance metrics. For more information, see [Configure monitoring for Auto Scaling instances \(p. 265\)](#).

To view monitoring graphs using the Amazon EC2 Auto Scaling console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to the Auto Scaling group that you want to view metrics for.
A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. Choose the **Monitoring** tab.
Amazon EC2 Auto Scaling displays monitoring graphs for **Auto Scaling** metrics.
4. To view monitoring graphs of the aggregated instance metrics for the group, choose **EC2**.

Graph actions

- Hover on a data point to view a data pop-up for a specific time in UTC.
- To enlarge a graph, choose **Enlarge** from the menu tool (the three vertical dots) in the upper right of the graph. Alternatively, choose the maximize icon at the top of the graph.
- Adjust the time period for data displayed in the graph by selecting one of the predefined time period values. If the graph is enlarged, you can choose **Custom** to define your own time period.
- Choose **Refresh** from the menu tool to update the data in a graph.
- Drag your cursor over the graph data to select a specific range. You can then choose **Apply time range** in the menu tool.
- Choose **View logs** from the menu tool to view associated log streams (if any) in the CloudWatch console.
- To view a graph in CloudWatch, choose **View in metrics** from the menu tool. This takes you to the CloudWatch page for that graph. There, you can view more information or access historical information to gain a better understanding of how your Auto Scaling group changed over an extended period.

Graph metrics for your Auto Scaling groups

After you create an Auto Scaling group, you can open the Amazon EC2 Auto Scaling console and view the monitoring graphs for the group on the **Monitoring** tab.

In the **Auto Scaling** section, the graph metrics include the following overall metrics. These metrics provide measurements that can be indicators of a potential issue, such as number of terminating instances or number of pending instances.

- **Minimum Group Size** (based on GroupMinSize)
- **Maximum Group Size** (based on GroupMaxSize)
- **Desired Capacity** (based on GroupDesiredCapacity)
- **In Service Instances** (based on GroupInServiceInstances)
- **Pending Instances** (based on GroupPendingInstances)
- **Standby Instances** (based on GroupStandbyInstances)
- **Terminating Instances** (based on GroupTerminatingInstances)
- **Total Instances** (based on GroupTotalInstances)

In the **EC2** section, you can find the following graph metrics based on key performance metrics for your Amazon EC2 instances. These EC2 metrics are an aggregate of metrics for all instances in the group.

- **CPU Utilization** (based on CPUUtilization)
- **Disk Reads** (based on DiskReadBytes)
- **Disk Read Operations** (based on DiskReadOps)
- **Disk Writes** (based on DiskWriteBytes)
- **Disk Write Operations** (based on DiskWriteOps)
- **Network In** (based on NetworkIn)
- **Network Out** (based on NetworkOut)
- **Status Check Failed (Any)** (based on StatusCheckFailed)
- **Status Check Failed (Instance)** (based on StatusCheckFailed_Instance)
- **Status Check Failed (System)** (based on StatusCheckFailed_System)

You can use the next set of **Auto Scaling** graph metrics to measure specific features of specific groups.

The following metric data is available for groups where instances have weights that define how many units each instance contributes to the desired capacity of the group:

- **In Service Capacity Units** (based on GroupInServiceCapacity)
- **Pending Capacity Units** (based on GroupPendingCapacity)
- **Standby Capacity Units** (based on GroupStandbyCapacity)
- **Terminating Capacity Units** (based on GroupTerminatingCapacity)
- **Total Capacity Units** (based on GroupTotalCapacity)

The following metric data is available if the group uses the [warm pool \(p. 219\)](#) feature:

- **Warm Pool Minimum Size** (based on WarmPoolMinSize)
- **Warm Pool Desired Capacity** (based on WarmPoolDesiredCapacity)
- **Warm Pool Pending Capacity Units** (based on WarmPoolPendingCapacity)
- **Warm Pool Terminating Capacity Units** (based on WarmPoolTerminatingCapacity)
- **Warm Pool Warmed Capacity Units** (based on WarmPoolWarmedCapacity)
- **Warm Pool Total Capacity Units Launched** (based on WarmPoolTotalCapacity)
- **Group and Warm Pool Desired Capacity** (based on GroupAndWarmPoolDesiredCapacity)

- **Group and Warm Pool Total Capacity Units Launched** (based on `GroupAndWarmPoolTotalCapacity`)

See also

For information about the data source for the **EC2** graph metrics, see [List the available CloudWatch metrics for your instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

To monitor per-instance metrics, see [Graph metrics for your instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Log Amazon EC2 Auto Scaling API calls with AWS CloudTrail

Amazon EC2 Auto Scaling is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or a service using Amazon EC2 Auto Scaling. CloudTrail captures all API calls for Amazon EC2 Auto Scaling as events. The calls captured include calls from the Amazon EC2 Auto Scaling console and code calls to the Amazon EC2 Auto Scaling API.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon EC2 Auto Scaling. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon EC2 Auto Scaling, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Amazon EC2 Auto Scaling information in CloudTrail

CloudTrail is enabled on your Amazon Web Services account when you create the account. When activity occurs in Amazon EC2 Auto Scaling, that activity is recorded in a CloudTrail event along with other Amazon Web Services events in **Event history**. You can view, search, and download recent events in your Amazon Web Services account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your Amazon Web Services account, including events for Amazon EC2 Auto Scaling, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all Regions. The trail logs events from all Regions in the Amazon Web Services partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other Amazon Web Services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All Amazon EC2 Auto Scaling actions are logged by CloudTrail and are documented in the [Amazon EC2 Auto Scaling API Reference](#). For example, calls to the **CreateLaunchConfiguration**, **DescribeAutoScalingGroup**, and **UpdateAutoScalingGroup** actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another service.

For more information, see the [CloudTrail userIdentity element](#).

Understand Amazon EC2 Auto Scaling log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the **CreateLaunchConfiguration** action.

```
{  
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "Root",  
    "principalId": "123456789012",  
    "arn": "arn:aws:iam::123456789012:root",  
    "accountId": "123456789012",  
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
    "sessionContext": {  
      "attributes": {  
        "mfaAuthenticated": "false",  
        "creationDate": "2018-08-21T17:05:42Z"  
      }  
    }  
  },  
  "eventTime": "2018-08-21T17:07:49Z",  
  "eventSource": "autoscaling.amazonaws.com",  
  "eventName": "CreateLaunchConfiguration",  
  "awsRegion": "us-west-2",  
  "sourceIPAddress": "192.0.2.0",  
  "userAgent": "Coral/Jakarta",  
  "requestParameters": {  
    "ebsOptimized": false,  
    "instanceMonitoring": {  
      "enabled": false  
    },  
    "instanceType": "t2.micro",  
    "keyName": "EC2-key-pair-oregon",  
    "blockDeviceMappings": [  
      {  
        "deviceName": "/dev/xvda",  
        "ebs": {  
          "deleteOnTermination": true,  
          "volumeSize": 8,  
          "snapshotId": "snap-01676e0a2c3c7de9e",  
          "volumeType": "gp2"  
        }  
      }  
    ],  
    "launchConfigurationName": "launch_configuration_1",  
  }  
}
```

```
        "imageId": "ami-6cd6f714d79675a5",
        "securityGroups": [
            "sg-00c429965fd921483"
        ],
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Get Amazon SNS notifications when your Auto Scaling group scales

You can be notified when Amazon EC2 Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. You manage notifications using Amazon Simple Notification Service (Amazon SNS).

Amazon SNS coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. Amazon SNS offers a variety of notification options, including the ability to deliver notifications as HTTP or HTTPS POST, email (SMTP, either plaintext or in JSON format), or as a message posted to an Amazon SQS queue, which enables you to handle these notifications programmatically. For more information, see [Amazon Simple Notification Service Developer Guide](#).

For example, if you configure your Auto Scaling group to use the `autoscaling:EC2_INSTANCE_TERMINATE` notification type, and your Auto Scaling group terminates an instance, it sends an email notification. This email contains the details of the terminated instance, such as the instance ID and the reason that the instance was terminated.

Notifications are useful for designing event-driven applications. If you use notifications to check that a resource enters a desired state, you can eliminate polling, and you won't encounter the `RequestLimitExceeded` error that sometimes results from polling.

AWS provides various tools that you can use to send notifications. Alternatively, you can use EventBridge and Amazon SNS to send notifications when your Auto Scaling groups launch or terminate instances. In EventBridge, the rule describes which events you're notified about. In Amazon SNS, the topic describes what kind of notification you receive. Using this option, you can decide if certain events should trigger a Lambda function instead. For more information, see [Use EventBridge to handle Auto Scaling events \(p. 304\)](#).

Contents

- [SNS notifications \(p. 271\)](#)
- [Configure Amazon SNS notifications for Amazon EC2 Auto Scaling \(p. 272\)](#)
 - [Create an Amazon SNS topic \(p. 272\)](#)
 - [Subscribe to the Amazon SNS topic \(p. 273\)](#)
 - [Confirm your Amazon SNS subscription \(p. 273\)](#)
 - [Configure your Auto Scaling group to send notifications \(p. 273\)](#)
 - [Test the notification \(p. 273\)](#)
 - [Delete the notification configuration \(p. 274\)](#)

SNS notifications

Amazon EC2 Auto Scaling supports sending Amazon SNS notifications when the following events occur.

Event	Description
autoscaling:EC2_INSTANCE_LAUNCH	Successful instance launch
autoscaling:EC2_INSTANCE_LAUNCH_ERROR	Failed instance launch
autoscaling:EC2_INSTANCE_TERMINATE	Successful instance termination
autoscaling:EC2_INSTANCE_TERMINATE_ERROR	Failed instance termination

The message includes the following information:

- Event — The event.
- AccountId — The Amazon Web Services account ID.
- AutoScalingGroupName — The name of the Auto Scaling group.
- AutoScalingGroupARN — The ARN of the Auto Scaling group.
- EC2InstanceId — The ID of the EC2 instance.

For example:

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
```

Configure Amazon SNS notifications for Amazon EC2 Auto Scaling

To use Amazon SNS to send email notifications, you must first create a *topic* and then subscribe your email addresses to the topic.

Create an Amazon SNS topic

An SNS topic is a logical access point, a communication channel your Auto Scaling group uses to send the notifications. You create a topic by specifying a name for your topic.

When you create a topic name, the name must meet the following requirements:

- Between 1 and 256 characters long
- Contain uppercase and lowercase ASCII letters, numbers, underscores, or hyphens

For more information, see [Creating an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Subscribe to the Amazon SNS topic

To receive the notifications that your Auto Scaling group sends to the topic, you must subscribe an endpoint to the topic. In this procedure, for **Endpoint**, specify the email address where you want to receive the notifications from Amazon EC2 Auto Scaling.

For more information, see [Subscribing to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Confirm your Amazon SNS subscription

Amazon SNS sends a confirmation email to the email address you specified in the previous step.

Make sure that you open the email from AWS Notifications and choose the link to confirm the subscription before you continue with the next step.

You will receive an acknowledgment message from AWS. Amazon SNS is now configured to receive notifications and send the notification as an email to the email address that you specified.

Configure your Auto Scaling group to send notifications

You can configure your Auto Scaling group to send notifications to Amazon SNS when a scaling event, such as launching instances or terminating instances, takes place. Amazon SNS sends a notification with information about the instances to the email address that you specified.

To configure Amazon SNS notifications for your Auto Scaling group (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.
A split pane opens up in the bottom part of the page, showing information about the group that's selected.
3. On the **Activity** tab, choose **Activity notifications**, **Create notification**.
4. On the **Create notifications** pane, do the following:
 - a. For **SNS Topic**, select your SNS topic.
 - b. For **Event types**, select the events to send the notifications.
 - c. Choose **Create**.

To configure Amazon SNS notifications for your Auto Scaling group (AWS CLI)

Use the following `put-notification-configuration` command.

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH" "autoscaling:EC2_INSTANCE_TERMINATE"
```

Test the notification

To generate a notification for a launch event, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. You receive a notification within a few minutes after instance launch.

To change the desired capacity (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.

A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Group details, Edit**.
4. For **Desired capacity**, increase the current value by 1. If this value exceeds **Maximum capacity**, you must also increase the value of **Maximum capacity** by 1.
5. Choose **Update**.
6. After a few minutes, you'll receive notification for the event. If you do not need the additional instance that you launched for this test, you can decrease **Desired capacity** by 1. After a few minutes, you'll receive notification for the event.

Delete the notification configuration

You can delete your Amazon EC2 Auto Scaling notification configuration if it is no longer being used.

To delete Amazon EC2 Auto Scaling notification configuration (console)

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select your Auto Scaling group.
3. On the **Activity** tab, select the check box next to the notification you want to delete and then choose **Actions, Delete**.

To delete Amazon EC2 Auto Scaling notification configuration (AWS CLI)

Use the following **delete-notification-configuration** command.

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --topic-arn arn
```

For information about deleting the Amazon SNS topic and all subscriptions associated with your Auto Scaling group, see [Deleting an Amazon SNS subscription and topic](#) in the *Amazon Simple Notification Service Developer Guide*.

AWS services integrated with Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling can be integrated with other AWS services. Review the following integration options to learn more about how each service works with Amazon EC2 Auto Scaling.

Topics

- [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#)
- [Create Auto Scaling groups with AWS CloudFormation \(p. 282\)](#)
- [Create Auto Scaling groups from the command line using AWS CloudShell \(p. 283\)](#)
- [Use AWS Compute Optimizer to get recommendations for the instance type for an Auto Scaling group \(p. 283\)](#)
- [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#)
- [Use EventBridge to handle Auto Scaling events \(p. 304\)](#)
- [Provide network connectivity for your Auto Scaling instances using Amazon VPC \(p. 317\)](#)

Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions

You can configure Amazon EC2 Auto Scaling to monitor and automatically respond to changes that affect the availability of your Spot Instances. Capacity Rebalancing helps you maintain workload availability by proactively augmenting your fleet with a new Spot Instance before a running instance is interrupted by Amazon EC2.

How it works

The goal of Capacity Rebalancing is to keep processing your workload without interruption. When Spot Instances are at an elevated risk of interruption, the Amazon EC2 Spot service notifies Amazon EC2 Auto Scaling with an EC2 instance rebalance recommendation.

When you enable Capacity Rebalancing for your Auto Scaling group, Amazon EC2 Auto Scaling attempts to proactively replace the Spot Instances in your group that have received a rebalance recommendation. This provides an opportunity to rebalance your workload to new Spot Instances that aren't at an elevated risk of interruption. Your workload can continue to process the work while Amazon EC2 Auto Scaling launches new Spot Instances before your existing instances are interrupted.

Optionally, you can also use a lifecycle hook to perform a custom action on the instances before they are terminated.

For more information about EC2 instance rebalance recommendations, see [EC2 instance rebalance recommendations](#) in the *Amazon EC2 User Guide for Linux Instances*.

For more details and a walkthrough of the Capacity Rebalancing feature, see the blog post [Proactively manage Spot Instance lifecycle using the new Capacity Rebalancing feature for EC2 Auto Scaling](#) on the AWS Compute Blog.

Note

When Capacity Rebalancing is disabled, Amazon EC2 Auto Scaling doesn't replace Spot Instances until after the Amazon EC2 Spot service interrupts the instances and their health

check fails. Before interrupting an instance, Amazon EC2 always gives both an EC2 instance rebalance recommendation and a Spot two-minute instance interruption notice.

Contents

- [Considerations \(p. 276\)](#)
 - [Capacity Rebalancing does not increase your Spot Instance interruption rate \(p. 277\)](#)
- [Enable Capacity Rebalancing \(p. 277\)](#)
 - [Enable Capacity Rebalancing \(console\) \(p. 277\)](#)
 - [Enable Capacity Rebalancing \(AWS CLI\) \(p. 278\)](#)
- [Add a termination lifecycle hook \(p. 281\)](#)

Considerations

If you configure an Auto Scaling group for Capacity Rebalancing, consider the following:

- We recommend that you configure your Auto Scaling group to use multiple instance types. This provides the flexibility to launch instances in various Spot Instance pools within each Availability Zone, as documented in [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).
- Your replacement Spot Instances may be at an elevated risk of interruption if you use the lowest-price allocation strategy. This is because we will always launch instances in the lowest priced pool that has available capacity at that moment, even if your replacement Spot Instances are likely to be interrupted soon after being launched. To avoid an elevated risk of interruption, we strongly recommend against using the lowest-price allocation strategy, and instead recommend the capacity-optimized or capacity-optimized-prioritized allocation strategy. These strategies aim to launch replacement Spot Instances in the most optimal Spot capacity pools so that they are less likely to be interrupted in the near future.
- Whenever possible, you should create your Auto Scaling group in all Availability Zones within the Region. This way, Amazon EC2 Auto Scaling can look at the available capacity in each Availability Zone. If a launch fails in one Availability Zone, Amazon EC2 Auto Scaling keeps trying to launch Spot Instances across the specified Availability Zones until it succeeds.
- With Capacity Rebalancing, Amazon EC2 Auto Scaling behaves in the following way:

When launching a new instance, Amazon EC2 Auto Scaling waits until the new instance passes its health check before it proceeds with terminating the previous instance. When replacing more than one instance, the termination of each previous instance starts after the new instance has launched and passed its health check. Because Amazon EC2 Auto Scaling attempts to launch new instances before terminating previous ones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities. To avoid this problem, Amazon EC2 Auto Scaling can temporarily exceed the specified maximum capacity of a group during a rebalancing activity.

- If the new instances fail to launch or they launch but the health check fails, Amazon EC2 Auto Scaling keeps trying to relaunch them. While it is trying to launch new instances, your previous ones will eventually be interrupted and forcibly terminated.
- If a scaling activity is in progress and your Auto Scaling group is below its new desired capacity, Amazon EC2 Auto Scaling scales out first before terminating the previous instances.
- You can configure a termination lifecycle hook for your Auto Scaling group when enabling Capacity Rebalancing to attempt a graceful shutdown of your application inside the instances that receive the rebalance notification, before Amazon EC2 Auto Scaling terminates the instances. If you don't configure a lifecycle hook, Amazon EC2 Auto Scaling starts terminating the previous instances as soon as the new instances pass their health check.
- Your application should be able to handle the possibility of a Spot Instance being interrupted early. For example, when an instance begins termination, Amazon EC2 Auto Scaling waits for the instance to terminate. If the Auto Scaling group is behind an Elastic Load Balancing load balancer, Amazon EC2 Auto Scaling waits for the instance to deregister from the load balancer before calling the termination

lifecycle hook (if configured). If the time to deregister the instance and complete lifecycle actions takes too long, the instance might be interrupted while Amazon EC2 Auto Scaling is waiting for the instance to terminate.

- It is also not always possible for Amazon EC2 to send the rebalance recommendation signal before the two-minute Spot Instance interruption notice. In some cases, the rebalance recommendation signal can arrive along with the two-minute interruption notice.
- In cases where an instance receives a final two-minute interruption notice, Amazon EC2 Auto Scaling calls the termination lifecycle hook and attempts to launch a replacement immediately.

Capacity Rebalancing does not increase your Spot Instance interruption rate

When you enable Capacity Rebalancing, it does not increase your [Spot Instance interruption rate](#) (the number of Spot Instances that are reclaimed when Amazon EC2 needs the capacity back). However, if Capacity Rebalancing detects an instance is at risk of interruption, Amazon EC2 Auto Scaling will immediately attempt to launch a new instance. The result is that more instances might be replaced than if you'd waited for Amazon EC2 Auto Scaling to launch a new instance after the at-risk instance was interrupted.

While you might replace more instances with Capacity Rebalancing enabled, you benefit from being proactive rather than reactive by having more time to take action before your instances are interrupted. With a [Spot Instance interruption notice](#), you typically only have up to two minutes to gracefully shut down your instance. With Capacity Rebalancing launching a new instance in advance, you give existing processes a better chance of completing on your at-risk instance, you can start your instance shutdown procedures, and prevent new work from being scheduled on your at-risk instance. You can also start preparing the newly launched instance to take over the application. With Capacity Rebalancing's proactive replacement, you benefit from graceful continuity.

As a theoretical example to demonstrate the risks and benefits of using Capacity Rebalancing, consider the following scenario:

- 2:00 PM – A rebalance recommendation is received for instance-A, and Amazon EC2 Auto Scaling immediately starts attempting to launch a replacement instance-B, giving you time to start your shutdown procedures.
- 2:30 PM – A rebalance recommendation is received for instance-B, replaced with instance-C, giving you time to start your shutdown procedures.
- 2:32 PM – If Capacity Rebalancing wasn't enabled, and if a Spot Instance interruption notice would've been received at 2:32 PM for instance-A, you would only have had up to two minutes to take action, but Instance-A would have been running up till this time.

Enable Capacity Rebalancing

You can enable or disable Capacity Rebalancing at any time.

Enable Capacity Rebalancing (console)

You can enable or disable Capacity Rebalancing when you create or update an Auto Scaling group.

To enable Capacity Rebalancing for a new Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Choose **Create Auto Scaling group**.

3. In step 1, enter a name for the Auto Scaling group, choose a launch template, and then choose **Next** to proceed to the next step.
4. For **Step 2: Choose instance launch options**, under **Network**, choose the options as desired.
5. For **Instance type requirements**, choose settings to create a mixed instances group, including the instance types that it can launch, instance purchase options, and allocation strategies for Spot and On-Demand Instances. By default, these settings are not configured. To configure them, you must select **Override launch template**. For more information about creating a mixed instances group, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).
6. Under the **Allocation strategies** section at the bottom of the page, choose a Spot allocation strategy. Enable or disable Capacity Rebalancing by selecting or clearing the **Capacity rebalance** check box. You only see this option if you specify a percentage of the Auto Scaling group to be launched as Spot Instances in the **Instance purchase options** section.
7. Create the Auto Scaling group.

To enable Capacity Rebalancing for an existing Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to your Auto Scaling group.
A split pane opens in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Allocation strategies**, **Edit**.
4. Under the **Allocation strategies** section, do the following:
 - To enable Capacity Rebalancing, select the **Capacity rebalance** check box.
 - To disable Capacity Rebalancing, clear the **Capacity rebalance** check box.
5. Choose **Update**.

Enable Capacity Rebalancing (AWS CLI)

The following examples show how to use the AWS CLI to enable and disable Capacity Rebalancing.

Use the [create-auto-scaling-group](#) or [update-auto-scaling-group](#) command with the following parameter:

- `--capacity-rebalance / --no-capacity-rebalance` — Boolean value that indicates whether Capacity Rebalancing is enabled.

Before you call the [create-auto-scaling-group](#) command, you need the name of a launch template that is configured for use with an Auto Scaling group. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

Note

The following procedures show how to use a configuration file formatted in JSON or YAML. If you use AWS CLI version 1, you must specify a JSON-formatted configuration file. If you use AWS CLI version 2, you can specify a configuration file formatted in either YAML or JSON.

JSON

To create and configure a new Auto Scaling group

- Use the following [create-auto-scaling-group](#) command to create a new Auto Scaling group and enable Capacity Rebalancing, referencing a JSON file as the sole parameter for your Auto Scaling group.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

If you don't already have a CLI configuration file that specifies a [mixed instances policy \(p. 52\)](#), create one.

Add the following line to the top-level JSON object in the configuration file.

```
{  
    "CapacityRebalance": true  
}
```

The following is an example config.json file.

```
{  
    "AutoScalingGroupName": "my-asg",  
    "DesiredCapacity": 12,  
    "MinSize": 12,  
    "MaxSize": 15,  
    "CapacityRebalance": true,  
    "MixedInstancesPolicy": {  
        "InstancesDistribution": {  
            "OnDemandBaseCapacity": 0,  
            "OnDemandPercentageAboveBaseCapacity": 25,  
            "SpotAllocationStrategy": "capacity-optimized"  
        },  
        "LaunchTemplate": {  
            "LaunchTemplateSpecification": {  
                "LaunchTemplateName": "my-launch-template",  
                "Version": "$Default"  
            },  
            "Overrides": [  
                {  
                    "InstanceType": "c5.large"  
                },  
                {  
                    "InstanceType": "c5a.large"  
                },  
                {  
                    "InstanceType": "m5.large"  
                },  
                {  
                    "InstanceType": "m5a.large"  
                },  
                {  
                    "InstanceType": "c4.large"  
                },  
                {  
                    "InstanceType": "m4.large"  
                },  
                {  
                    "InstanceType": "c3.large"  
                },  
                {  
                    "InstanceType": "m3.large"  
                }  
            ]  
        }  
    },  
    "TargetGroupARNs": "arn:aws:elasticloadbalancing:us-  
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becfff",  
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

}

YAML

To create and configure a new Auto Scaling group

- Use the following [create-auto-scaling-group](#) command to create a new Auto Scaling group and enable Capacity Rebalancing, referencing a YAML file as the sole parameter for your Auto Scaling group.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

Add the following line to your configuration file formatted in YAML.

```
CapacityRebalance: true
```

The following is an example config.yaml file.

```
---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
  Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  TargetGroupARNs:
    - arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

To enable Capacity Rebalancing for an existing Auto Scaling group

- Use the following [update-auto-scaling-group](#) command to enable Capacity Rebalancing.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

To verify that Capacity Rebalancing is enabled for an Auto Scaling group

- Use the following [describe-auto-scaling-groups](#) command to verify that Capacity Rebalancing is enabled and to view the details.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

The following is an example response.

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "my-asg",  
      "AutoScalingGroupARN": "arn",  
      ...  
      "CapacityRebalance": true  
    }  
  ]  
}
```

To disable Capacity Rebalancing

Use the [update-auto-scaling-group](#) command with the `--no-capacity-rebalance` option to disable Capacity Rebalancing.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
  --no-capacity-rebalance
```

Add a termination lifecycle hook

Optionally, configure a termination lifecycle hook when you enable Capacity Rebalancing. With a lifecycle hook, you can perform custom actions on an instance before it is terminated.

The following are some reasons why you might use a termination lifecycle hook:

- To upload system or application logs to Amazon Simple Storage Service (Amazon S3)
- For graceful shutdown of Amazon SQS workers
- To complete deregistration from the Domain Name System (DNS)

If you don't have a termination lifecycle hook, use the following procedure to create one.

To add a termination lifecycle hook

- Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
- Select the check box next to your Auto Scaling group.

A split pane opens in the bottom of the **Auto Scaling groups** page.

- On the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.
- To define a lifecycle hook, do the following:
 - For **Lifecycle hook name**, specify a name for the lifecycle hook.
 - For **Lifecycle transition**, choose **Instance terminate**.

- c. For **Heartbeat timeout**, specify the amount of time, in seconds, that you will have to complete the lifecycle action, or until the timeout period ends. We recommend a value from 30 to 120 seconds, depending on how much time you need to shut down your application.
 - d. For **Default result**, specify the action that the Auto Scaling group takes when the timeout elapses or if an unexpected failure occurs. Both **ABANDON** and **CONTINUE** let the instance terminate.
 - If you choose **CONTINUE**, the Auto Scaling group can proceed with any remaining actions, such as other lifecycle hooks, before termination.
 - If you choose **ABANDON**, the Auto Scaling group terminates the instances immediately.
 - e. (Optional) For **Notification metadata**, specify additional information that you want to include anytime that Amazon EC2 Auto Scaling sends a message to an AWS Lambda function or another notification target that you configure in step 6.
5. Choose **Create**.
 6. (Optional) To use a service such as Lambda to perform a custom action before instance termination, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#). This tutorial will help you understand how to set up a Lambda function for your lifecycle hook. Otherwise, for an EC2 instance to run an action automatically, you must configure it to run a shutdown script. We recommend that you script your entire shutdown sequence to be completed in under one to two minutes to make sure that there is enough time to complete tasks before instance termination.

For more information to help you understand different aspects of working with lifecycle hooks, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Note

Support for Capacity Rebalancing is available in all commercial AWS Regions where Amazon EC2 Auto Scaling is available, excluding the Middle East (UAE) Region.

Create Auto Scaling groups with AWS CloudFormation

Amazon EC2 Auto Scaling is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as Auto Scaling groups), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your Amazon EC2 Auto Scaling resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Amazon EC2 Auto Scaling and AWS CloudFormation templates

To provision and configure resources for Amazon EC2 Auto Scaling and related services, you must understand [AWS CloudFormation templates](#). Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see [What is AWS CloudFormation Designer?](#) in the [AWS CloudFormation User Guide](#).

Amazon EC2 Auto Scaling supports creating Auto Scaling groups and launch configurations in AWS CloudFormation. For more information, including examples of JSON and YAML templates for Auto

Scaling groups, see the [Amazon EC2 Auto Scaling resource type reference](#) in the *AWS CloudFormation User Guide*.

You can find additional useful examples of templates that create Auto Scaling groups and related resources in the following sections of the *AWS CloudFormation User Guide*.

- For various examples of Amazon EC2 launch templates, see [AWS::EC2::LaunchTemplate](#).
- For examples that cover both Amazon EC2 launch templates and Auto Scaling groups, see [Auto scaling template snippets](#).

Learn more about AWS CloudFormation

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

- [AWS CloudFormation](#)
- [AWS CloudFormation User Guide](#)
- [AWS CloudFormation API Reference](#)
- [AWS CloudFormation Command Line Interface User Guide](#)

Create Auto Scaling groups from the command line using AWS CloudShell

In [supported AWS Regions](#), you can run AWS CLI commands using AWS CloudShell for a browser-based, pre-authenticated shell that launches directly from the AWS Management Console. You can run AWS CLI commands against services using your preferred shell (Bash, PowerShell, or Z shell).

You can launch AWS CloudShell from the AWS Management Console using either one of the following two methods:

- Choose the AWS CloudShell icon on the console navigation bar. It's located to the right of the search box.
- Use the search box on the console navigation bar to search for **CloudShell** and then choose the **CloudShell** option.

When AWS CloudShell launches in a new browser window for the first time, a welcome panel displays and lists key features. After you close this panel, status updates are provided while the shell configures and forwards your console credentials. When the command prompt displays, the shell is ready for interaction.

For more information on this service, see the [AWS CloudShell User Guide](#).

Use AWS Compute Optimizer to get recommendations for the instance type for an Auto Scaling group

AWS provides Amazon EC2 instance recommendations to help you improve performance, save money, or both, by using features powered by AWS Compute Optimizer. You can use these recommendations to decide whether to move to a new instance type.

To make recommendations, Compute Optimizer analyzes your existing instance specifications and recent metric history. The compiled data is then used to recommend which Amazon EC2 instance types are best optimized to handle the existing performance workload. Recommendations are returned along with per-hour instance pricing.

Note

To get recommendations from Compute Optimizer, you must first opt in to Compute Optimizer. For more information, see [Getting started with AWS Compute Optimizer](#) in the *AWS Compute Optimizer User Guide*.

Contents

- [Limitations \(p. 284\)](#)
- [Findings \(p. 284\)](#)
- [View recommendations \(p. 284\)](#)
- [Considerations for evaluating the recommendations \(p. 285\)](#)

Limitations

Compute Optimizer generates recommendations for instances in Auto Scaling groups that are configured to launch and run M, C, R, T, and X instance types. However, it does not generate recommendations for -g instance types powered by AWS Graviton2 processors (e.g., C6g), and for -n instance types that have higher network bandwidth performance (e.g., M5n).

The Auto Scaling groups must also be configured to run a single instance type (i.e., no mixed instance types), must not have a scaling policy attached to them, and have the same values for desired, minimum, and maximum capacity (i.e., an Auto Scaling group with a fixed number of instances). Compute Optimizer generates recommendations for instances in Auto Scaling groups that meet *all* of these configuration requirements.

Findings

Compute Optimizer classifies its findings for Auto Scaling groups as follows:

- **Not optimized** – An Auto Scaling group is considered not optimized when Compute Optimizer has identified a recommendation that can provide better performance for your workload.
- **Optimized** – An Auto Scaling group is considered optimized when Compute Optimizer determines that the group is correctly provisioned to run your workload, based on the chosen instance type. For optimized resources, Compute Optimizer might sometimes recommend a new generation instance type.
- **None** – There are no recommendations for this Auto Scaling group. This might occur if you've been opted in to Compute Optimizer for less than 12 hours, or when the Auto Scaling group has been running for less than 30 hours, or when the Auto Scaling group or instance type is not supported by Compute Optimizer. For more information, see the [Limitations \(p. 284\)](#) section.

View recommendations

After you opt in to Compute Optimizer, you can view the findings and recommendations that it generates for your Auto Scaling groups. If you recently opted in, recommendations might not be available for up to 12 hours.

To view recommendations generated for an Auto Scaling group

1. Open the Compute Optimizer console at <https://console.aws.amazon.com/compute-optimizer/>.

The Dashboard page opens.

2. Choose **View recommendations for all Auto Scaling groups**.
3. Select your Auto Scaling group.
4. Choose **View detail**.

The view changes to display up to three different instance recommendations in a preconfigured view, based on default table settings. It also provides recent CloudWatch metric data (average CPU utilization, average network in, and average network out) for the Auto Scaling group.

Determine whether you want to use one of the recommendations. Decide whether to optimize for performance improvement, for cost reduction, or for a combination of these two.

To change the instance type in your Auto Scaling group, update the launch template or update the Auto Scaling group to use a new launch configuration. Existing instances continue to use the previous configuration. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow automatic scaling to gradually replace older instances with newer instances based on your [termination policies \(p. 230\)](#).

Note

With the maximum instance lifetime and instance refresh features, you can also replace existing instances in your Auto Scaling group to launch new instances that use the new launch template or launch configuration. For more information, see [Replace Auto Scaling instances based on maximum instance lifetime \(p. 114\)](#) and [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Considerations for evaluating the recommendations

Before moving to a new instance type, consider the following:

- The recommendations don't forecast your usage. Recommendations are based on your historical usage over the most recent 14-day time period. Be sure to choose an instance type that is expected to meet your future usage needs.
- Focus on the graphed metrics to determine whether actual usage is lower than instance capacity. You can also view metric data (average, peak, percentile) in CloudWatch to further evaluate your EC2 instance recommendations. For example, notice how CPU percentage metrics change during the day and whether there are peaks that need to be accommodated. For more information, see [Viewing available metrics](#) in the *Amazon CloudWatch User Guide*.
- Compute Optimizer might supply recommendations for burstable performance instances, which are T3, T3a, and T2 instances. If you periodically burst above your baseline, make sure that you can continue to do so based on the vCPUs of the new instance type. For more information, see [CPU credits and baseline performance for burstable performance instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
- If you've purchased a Reserved Instance, your On-Demand Instance might be billed as a Reserved Instance. Before you change your current instance type, first evaluate the impact on Reserved Instance utilization and coverage.
- Consider conversions to newer generation instances, where possible.
- When migrating to a different instance family, make sure the current instance type and the new instance type are compatible, for example, in terms of virtualization, architecture, or network type. For more information, see [Compatibility for resizing instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Finally, consider the performance risk rating that's provided for each recommendation. Performance risk indicates the amount of effort you might need to spend in order to validate whether the recommended instance type meets the performance requirements of your workload. We also recommend rigorous load and performance testing before and after making any changes.

Additional resources

In addition to the topics on this page, see the following resources:

- [Amazon EC2 Instance Types](#)
- [AWS Compute Optimizer User Guide](#)

Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group

Elastic Load Balancing automatically distributes your incoming application traffic across all the EC2 instances that you are running. Elastic Load Balancing helps to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed.

To use Elastic Load Balancing with your Auto Scaling group, [attach the load balancer to your Auto Scaling group \(p. 288\)](#). This registers the group with the load balancer, which acts as a single point of contact for all incoming web traffic to your Auto Scaling group.

When you use Elastic Load Balancing with your Auto Scaling group, it's not necessary to register individual EC2 instances with the load balancer. Instances that are launched by your Auto Scaling group are automatically registered with the load balancer. Likewise, instances that are terminated by your Auto Scaling group are automatically deregistered from the load balancer.

After attaching a load balancer to your Auto Scaling group, you can configure your Auto Scaling group to use Elastic Load Balancing metrics (such as the Application Load Balancer request count per target) to scale the number of instances in the group as demand fluctuates.

Optionally, you can add Elastic Load Balancing health checks to your Auto Scaling group so that Amazon EC2 Auto Scaling can identify and replace unhealthy instances based on these additional health checks. Otherwise, you can create a CloudWatch alarm that notifies you if the healthy host count of the target group is lower than allowed.

Contents

- [Elastic Load Balancing types \(p. 286\)](#)
- [Prerequisites for getting started with Elastic Load Balancing \(p. 287\)](#)
- [Attach a load balancer to your Auto Scaling group \(p. 288\)](#)
- [Understand the attachment status of your load balancer \(p. 290\)](#)
- [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#)
- [Add Elastic Load Balancing health checks to an Auto Scaling group \(p. 291\)](#)
- [Add and remove Availability Zones \(p. 292\)](#)
- [Examples for working with Elastic Load Balancing with the AWS Command Line Interface \(AWS CLI\) \(p. 294\)](#)
- [Tutorial: Set up a scaled and load-balanced application \(p. 297\)](#)

Elastic Load Balancing types

Elastic Load Balancing provides four types of load balancers that can be used with your Auto Scaling group: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers.

There is a key difference in how the load balancer types are configured. With Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, instances are registered as targets with a target group, and you route traffic to the target group. With Classic Load Balancers, instances are registered directly with the load balancer.

Application Load Balancer

Routes and load balances at the application layer (HTTP/HTTPS), and supports path-based routing. An Application Load Balancer can route requests to ports on one or more registered targets, such as EC2 instances, in your virtual private cloud (VPC).

Network Load Balancer

Routes and load balances at the transport layer (TCP/UDP Layer-4), based on address information extracted from the Layer-4 header. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.

Gateway Load Balancer

Distributes traffic to a fleet of appliance instances. Provides scale, availability, and simplicity for third-party virtual appliances, such as firewalls, intrusion detection and prevention systems, and other appliances. Gateway Load Balancers work with virtual appliances that support the GENEVE protocol. Additional technical integration is required, so make sure to consult the user guide before choosing a Gateway Load Balancer.

Classic Load Balancer

Routes and load balances either at the transport layer (TCP/SSL), or at the application layer (HTTP/HTTPS).

To learn more about Elastic Load Balancing, see the following topics:

- [What is Elastic Load Balancing?](#)
- [What is an Application Load Balancer?](#)
- [What is a Network Load Balancer?](#)
- [What is a Gateway Load Balancer?](#)
- [What is a Classic Load Balancer?](#)

Prerequisites for getting started with Elastic Load Balancing

Follow the procedures in the Elastic Load Balancing documentation to create the load balancer and target group. Skip the step for registering your Amazon EC2 instances. Amazon EC2 Auto Scaling automatically takes care of registering (and deregistering) instances. For more information, see [Getting started with Elastic Load Balancing](#) in the *Elastic Load Balancing User Guide*.

Alternatively, if you want to create an Application Load Balancer or Network Load Balancer, you do not need to create the load balancer and target group now. You can create and attach a new Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console. For more information, see [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#).

To attach a load balancer to your Auto Scaling group, first make sure you have met these prerequisites:

- The load balancer and its target group must be in the same VPC and in the same Region as your Auto Scaling group.

- The target group must specify a target type of `instance`. You can't specify a target type of `ip` when using an Auto Scaling group.
- The security group for your launch template or launch configuration must be set up to allow traffic from the load balancer to reach your Auto Scaling group. The recommended rules depend on the type of load balancer and the types of backends that the load balancer uses. For example, to route traffic to web servers, allow inbound HTTP access on port 80 from the load balancer.
- When deploying virtual appliances behind a Gateway Load Balancer, the Amazon Machine Image (AMI) must specify the ID of an AMI that supports the GENEVE protocol to allow the Auto Scaling group to exchange traffic with a Gateway Load Balancer. Also, the security groups that you specify in the launch template or launch configuration must allow UDP traffic on port 6081.

Before allowing your Auto Scaling group to use the results of the Elastic Load Balancing health checks to determine an instance's health, make sure you have met this additional prerequisite:

- The security groups that you specify in the launch template or launch configuration must allow access from the load balancer on the correct port for Elastic Load Balancing to perform its health checks. For more information, see [Add Elastic Load Balancing health checks to an Auto Scaling group \(p. 291\)](#).

Tip

If you have bootstrapping scripts that take a while to complete, you can optionally add a launch lifecycle hook to your Auto Scaling group to delay instances from being registered behind the load balancer before your bootstrap scripts have completed successfully and the applications on the instances are ready to accept traffic. You can't add a lifecycle hook when you initially create an Auto Scaling group in the Amazon EC2 Auto Scaling console. You can add a lifecycle hook after the group is created. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

Attach a load balancer to your Auto Scaling group

This topic describes how to attach an Elastic Load Balancing load balancer to your Auto Scaling group. Amazon EC2 Auto Scaling integrates with Elastic Load Balancing to help you to insert an Application Load Balancer, Network Load Balancer, Classic Load Balancer, or Gateway Load Balancer in front of your Auto Scaling group. To learn more about the different types of load balancers, see [Elastic Load Balancing types \(p. 286\)](#).

When you attach an Application Load Balancer, Network Load Balancer, or Gateway Load Balancer, you attach a target group. Amazon EC2 Auto Scaling adds instances to the attached target group when they are launched. You can attach one or multiple target groups, and configure health checks on a per target group basis.

For an introductory guide for attaching a target group to your Auto Scaling group, see [Tutorial: Set up a scaled and load-balanced application \(p. 297\)](#).

Contents

- [Attach an existing load balancer \(p. 288\)](#)
- [Detach a load balancer \(p. 289\)](#)

Attach an existing load balancer

You can attach an existing load balancer to an Auto Scaling group when you create or update the group. If you want to create and attach a new Application Load Balancer or Network Load Balancer at the same time that you create the group, see [Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console \(p. 290\)](#).

To attach an existing load balancer as you are creating a new Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Choose **Create Auto Scaling group**.
3. In steps 1 and 2, choose the options as desired and proceed to **Step 3: Configure advanced options**.
4. For **Load balancing**, choose **Attach to an existing load balancer**.
5. Under **Attach to an existing load balancer**, do one of the following:
 - a. For Application Load Balancers, Network Load Balancers, and Gateway Load Balancers:
Choose **Choose from your load balancer target groups**, and then choose a target group in the **Existing load balancer target groups** field.
 - b. For Classic Load Balancers:
Choose **Choose from Classic Load Balancers**, and then choose your load balancer in the **Classic Load Balancers** field.
6. Proceed to create the Auto Scaling group. Your instances will be automatically registered to the load balancer after the Auto Scaling group has been created.

To attach an existing load balancer to an existing Auto Scaling group

Use the following procedure to attach a load balancer to an existing Auto Scaling group.

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Load balancing, Edit**.
4. Under **Load balancing**, do one of the following:
 - a. For **Application, Network or Gateway Load Balancer target groups**, select its check box and choose a target group.
 - b. For **Classic Load Balancers**, select its check box and choose your load balancer.
5. Choose **Update**.

Detach a load balancer

When you no longer need the load balancer, use the following procedure to detach it from your Auto Scaling group.

To detach a load balancer from a group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Load balancing, Edit**.
4. Under **Load balancing**, do one of the following:
 - a. For **Application, Network or Gateway Load Balancer target groups**, choose the delete (X) icon next to the target group.

- b. For **Classic Load Balancers**, choose the delete (X) icon next to the load balancer.
5. Choose **Update**.

Understand the attachment status of your load balancer

When you describe target groups using the [describe-load-balancer-target-groups](#) command or load balancers using the [describe-load-balancers](#) command, it returns the attachment status of the load balancer in the `State` parameter.

When you attach a load balancer, it enters the `Adding` state while registering the instances in the group. After all instances in the group are registered, it enters the `Added` state. After at least one registered instance passes the health checks, it enters the `InService` state. When the load balancer is in the `InService` state, Amazon EC2 Auto Scaling can terminate and replace any instances that are reported as unhealthy. If no registered instances pass the health checks (for example, due to a misconfigured health check), the load balancer doesn't enter the `InService` state. Amazon EC2 Auto Scaling doesn't terminate and replace the instances.

When you detach a load balancer, it enters the `Removing` state while deregistering the instances in the group. The instances remain running after they are deregistered. By default, connection draining (deregistration delay) is enabled for Application Load Balancers, Network Load Balancers, and Gateway Load Balancers. If connection draining is enabled, Elastic Load Balancing waits for in-flight requests to complete or for the maximum timeout to expire (whichever comes first) before it deregisters the instances.

Configure an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console

Use the following procedure to create and attach an Application Load Balancer or a Network Load Balancer as you create your Auto Scaling group.

To create and attach a new load balancer as you create a new Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Choose **Create Auto Scaling group**.
3. In steps 1 and 2, choose the options as desired and proceed to **Step 3: Configure advanced options**.
4. For **Load balancing**, choose **Attach to a new load balancer**.
 - a. Under **Attach to a new load balancer**, for **Load balancer type**, choose whether to create an Application Load Balancer or Network Load Balancer.
 - b. For **Load balancer name**, enter a name for the load balancer, or keep the default name.
 - c. For **Load balancer scheme**, choose whether to create a public internet-facing load balancer, or keep the default for an internal load balancer.
 - d. For **Availability Zones and subnets**, select the public subnet for each Availability Zone in which you chose to launch your EC2 instances. (These prepopulate from step 2.).
 - e. For **Listeners and routing**, update the port number for your listener (if necessary), and under **Default routing**, choose **Create a target group**. Alternatively, you can choose an existing target group from the drop-down list.

- f. If you chose **Create a target group** in the last step, for **New target group name**, enter a name for the target group, or keep the default name.
 - g. To add tags to your load balancer, choose **Add tag**, and provide a tag key and value for each tag.
5. Proceed to create the Auto Scaling group. Your instances will be automatically registered to the load balancer after the Auto Scaling group has been created.

Note

After creating your Auto Scaling group, you can use the Elastic Load Balancing console to create additional listeners. This is useful if you need to create a listener with a secure protocol, such as HTTPS, or a UDP listener. You can add more listeners to existing load balancers, as long as you use distinct ports.

Add Elastic Load Balancing health checks to an Auto Scaling group

The default health checks for an Auto Scaling group are EC2 status checks only. If an instance fails these status checks, it is marked unhealthy and is terminated while Amazon EC2 Auto Scaling launches a new replacement instance.

You can attach one or more load balancer target groups, one or more Classic Load Balancers, or both to your Auto Scaling group. However, by default, the Auto Scaling group does not consider an instance unhealthy and replace it if it fails the Elastic Load Balancing health checks.

To ensure that your Auto Scaling group can determine instance health based on additional load balancer tests, configure the Auto Scaling group to use Elastic Load Balancing (ELB) health checks. The load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances and determines if an instance is unhealthy. If you configure the Auto Scaling group to use Elastic Load Balancing health checks, it considers the instance unhealthy if it fails either the EC2 status checks or the Elastic Load Balancing health checks. If you attach multiple load balancer target groups or Classic Load Balancers to the group, all of them must report that an instance is healthy in order for it to consider the instance healthy. If any one of them reports an instance as unhealthy, the Auto Scaling group replaces the instance, even if others report it as healthy.

Add Elastic Load Balancing health checks

Use the following procedure to add Elastic Load Balancing (ELB) health checks to an Auto Scaling group.

To add Elastic Load Balancing health checks

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
 2. Select the check box next to an existing group.
- A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Health checks, Edit**.
 4. For **Health check type**, select **Enable ELB health checks**.
 5. For **Health check grace period**, enter the amount of time, in seconds, that Amazon EC2 Auto Scaling needs to wait before checking the health status of an instance after it enters the **InService** state. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).
 6. Choose **Update**.
 7. On the **Instance management** tab, under **Instances**, you can view the health status of instances. The **Health Status** column displays the results of the newly added health checks.

See also

- For more information, see [Health checks for Auto Scaling instances \(p. 253\)](#).
- To configure health checks for your Application Load Balancer, see [Health checks for your target groups](#) in the *User Guide for Application Load Balancers*.
- To configure health checks for your Network Load Balancer, see [Health checks for your target groups](#) in the *User Guide for Network Load Balancers*.
- To configure health checks for your Gateway Load Balancer, see [Health checks for your target groups](#) in the *User Guide for Gateway Load Balancers*.
- To configure health checks for your Classic Load Balancer, see [Configure health checks for your Classic Load Balancer](#) in the *User Guide for Classic Load Balancers*.

Add and remove Availability Zones

To take advantage of the safety and reliability of geographic redundancy, span your Auto Scaling group across multiple Availability Zones within a Region and attach a load balancer to distribute incoming traffic across those Availability Zones.

When one Availability Zone becomes unhealthy or unavailable, Amazon EC2 Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Amazon EC2 Auto Scaling automatically redistributes the application instances evenly across all the Availability Zones for your Auto Scaling group. Amazon EC2 Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Amazon EC2 Auto Scaling attempts to launch in other Availability Zones until it succeeds.

Elastic Load Balancing creates a load balancer node for each Availability Zone you enable for the load balancer. If you enable cross-zone load balancing for your load balancer, each load balancer node distributes traffic evenly across the registered instances in all enabled Availability Zones. If cross-zone load balancing is disabled, each load balancer node distributes requests evenly across the registered instances in its Availability Zone only.

You must specify at least one Availability Zone when you are creating your Auto Scaling group. Later, you can expand the availability of your application by adding an Availability Zone to your Auto Scaling group and enabling that Availability Zone for your load balancer (if the load balancer supports it).

Contents

- [Add an Availability Zone \(p. 292\)](#)
- [Remove an Availability Zone \(p. 293\)](#)
- [Limitations \(p. 294\)](#)

Add an Availability Zone

Use the following procedure to expand your Auto Scaling group and load balancer to a subnet in an additional Availability Zone.

To add an Availability Zone

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.
A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Network, Edit**.

4. In **Subnets**, choose the subnet corresponding to the Availability Zone that you want to add to the Auto Scaling group.
5. Choose **Update**.
6. To update the Availability Zones for your load balancer so that it shares the same Availability Zones as your Auto Scaling group, complete the following steps:
 - a. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
 - b. Choose your load balancer.
 - c. Do one of the following:
 - For Application Load Balancers and Network Load Balancers:
 1. On the **Description** tab, for **Availability Zones**, choose **Edit subnets**.
 2. On the **Edit subnets** page, for **Availability Zones**, select the check box for the Availability Zone to add. If there is only one subnet for that zone, it is selected. If there is more than one subnet for that zone, select one of the subnets.
 - For Classic Load Balancers in a VPC:
 1. On the **Instances** tab, choose **Edit Availability Zones**.
 2. On the **Add and Remove Subnets** page, for **Available subnets**, select the subnet using its add (+) icon. The subnet is moved under **Selected subnets**.
 - d. Choose **Save**.

Remove an Availability Zone

To remove an Availability Zone from your Auto Scaling group and load balancer, use the following procedure.

To remove an Availability Zone

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>, and choose **Auto Scaling Groups** from the navigation pane.
2. Select the check box next to an existing group.

A split pane opens up in the bottom of the **Auto Scaling groups** page.
3. On the **Details** tab, choose **Network, Edit**.
4. In **Subnets**, choose the delete (X) icon for the subnet corresponding to the Availability Zone that you want to remove from the Auto Scaling group. If there is more than one subnet for that zone, choose the delete (X) icon for each one.
5. Choose **Update**.
6. To update the Availability Zones for your load balancer so that it shares the same Availability Zones as your Auto Scaling group, complete the following steps:
 - a. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
 - b. Choose your load balancer.
 - c. Do one of the following:
 - For Application Load Balancers and Network Load Balancers:
 1. On the **Description** tab, for **Availability Zones**, choose **Edit subnets**.
 2. On the **Edit subnets** page, for **Availability Zones**, clear the check box to remove the subnet for that Availability Zone.
 - For Classic Load Balancers in a VPC:
 1. On the **Instances** tab, choose **Edit Availability Zones**.

2. On the **Add and Remove Subnets** page, for **Available subnets**, remove the subnet using its delete (-) icon. The subnet is moved under **Available subnets**.

d. Choose **Save**.

Limitations

To update which Availability Zones are enabled for your load balancer, you need to be aware of the following limitations:

- When you enable an Availability Zone for your load balancer, you specify one subnet from that Availability Zone. Note that you can enable at most one subnet per Availability Zone for your load balancer.
- For internet-facing load balancers, the subnets that you specify for the load balancer must have at least eight available IP addresses.
- For Application Load Balancers, you must enable at least two Availability Zones.
- For Network Load Balancers, you cannot disable the enabled Availability Zones, but you can enable additional ones.
- For Gateway Load Balancers, you cannot change the Availability Zones or subnets that were added when the load balancer was created.

Examples for working with Elastic Load Balancing with the AWS Command Line Interface (AWS CLI)

Use the AWS CLI to attach and detach load balancers, add Elastic Load Balancing health checks, and update Availability Zones.

Contents

- [Attach a load balancer target group \(p. 294\)](#)
- [Describe load balancer target groups \(p. 295\)](#)
- [Detach a load balancer target group \(p. 295\)](#)
- [Attach a Classic Load Balancer \(p. 295\)](#)
- [Describe Classic Load Balancers \(p. 295\)](#)
- [Detach a Classic Load Balancer \(p. 296\)](#)
- [Add Elastic Load Balancing health checks \(p. 296\)](#)
- [Update Availability Zones \(p. 296\)](#)

Attach a load balancer target group

The following `create-auto-scaling-group` command creates an Auto Scaling group with an attached target group. Specify the Amazon Resource Name (ARN) of a target group for an Application Load Balancer, Network Load Balancer, or Gateway Load Balancer.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
--launch-template "LaunchTemplateName=my-launch-template,Version=1" \
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
--target-group-arns "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-targets/1234567890123456" \
--max-size 5 --min-size 1 --desired-capacity 2
```

The following [attach-load-balancer-target-groups](#) command attaches a target group to an existing Auto Scaling group.

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \  
--target-group-arns "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-  
targets/1234567890123456"
```

Describe load balancer target groups

To view the target groups associated with an Auto Scaling group, use the [describe-load-balancer-target-groups](#) command. The following example lists the target groups for *my-asg*.

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

For an explanation of the State field in the output, see the [Understand the attachment status of your load balancer \(p. 290\)](#) section in a previous topic.

Detach a load balancer target group

The following [detach-load-balancer-target-groups](#) command detaches a target group from your Auto Scaling group when you no longer need it.

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \  
--target-group-arns "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-  
targets/1234567890123456"
```

Attach a Classic Load Balancer

The following [create-auto-scaling-group](#) command creates an Auto Scaling group with an attached Classic Load Balancer.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \  
--launch-configuration-name my-launch-config \  
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \  
--load-balancer-names "my-load-balancer" \  
--max-size 5 --min-size 1 --desired-capacity 2
```

The following [attach-load-balancers](#) command attaches the specified Classic Load Balancer to an existing Auto Scaling group.

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \  
--load-balancer-names my-lb
```

Describe Classic Load Balancers

To view the Classic Load Balancers associated with an Auto Scaling group, use the [describe-load-balancers](#) command. The following example lists the Classic Load Balancers for *my-asg*.

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

For an explanation of the State field in the output, see [Understand the attachment status of your load balancer \(p. 290\)](#).

Detach a Classic Load Balancer

The following [detach-load-balancers](#) command detaches a Classic Load Balancer from your Auto Scaling group when you no longer need it.

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \  
--load-balancer-names my-lb
```

Add Elastic Load Balancing health checks

To add Elastic Load Balancing health checks to an Auto Scaling group, run the following [update-auto-scaling-group](#) command and specify ELB as the value for the `--health-check-type` option.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-lb-asg \  
--health-check-type ELB
```

To update the health check grace period, use the `--health-check-grace-period` option. New instances often need time for a brief warm-up before they can pass a health check. If the grace period doesn't provide enough warm-up time, the instances might not appear ready to serve traffic. Amazon EC2 Auto Scaling might consider those instances unhealthy and replace them. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).

The following [update-auto-scaling-group](#) command adds Elastic Load Balancing health checks and specifies a grace period of 300 seconds.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-lb-asg \  
--health-check-type ELB --health-check-grace-period 300
```

Update Availability Zones

The commands that you use depend on whether your load balancer is an Application Load Balancer or Network Load Balancer, or a Classic Load Balancer. You can update the subnets and Availability Zones for your load balancer only if your load balancer supports it. For more information, see [Limitations \(p. 294\)](#).

For an Auto Scaling group with an Application Load Balancer or Network Load Balancer

1. Specify the subnets that are used for the Auto Scaling group using the following [update-auto-scaling-group](#) command.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
--vpc-zone-identifier subnet-41767929 subnet-cb663da2 subnet-8360a9e7
```

2. Verify that the instances in the new subnets are ready to accept traffic from the load balancer using the following [describe-auto-scaling-groups](#) command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. Specify the subnets that are used for your Application Load Balancer or Network Load Balancer using the following [set-subnets](#) command.

```
aws elbv2 set-subnets --load-balancer-arn my-lb-arn \  
--subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
```

For an Auto Scaling group with a Classic Load Balancer in a VPC

1. Specify the subnets that are used for the Auto Scaling group using the following [update-auto-scaling-group](#) command.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \  
  --vpc-zone-identifier subnet-41767929 subnet-cb663da2
```

2. Verify that the instances in the new subnets are ready to accept traffic from the load balancer using the following [describe-auto-scaling-groups](#) command.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. Enable the new subnet for your Classic Load Balancer using the following [attach-load-balancer-to-subnets](#) command.

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \  
  --subnets subnet-cb663da2
```

To disable a subnet, run the following [detach-load-balancer-from-subnets](#) command.

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \  
  --subnets subnet-8360a9e7
```

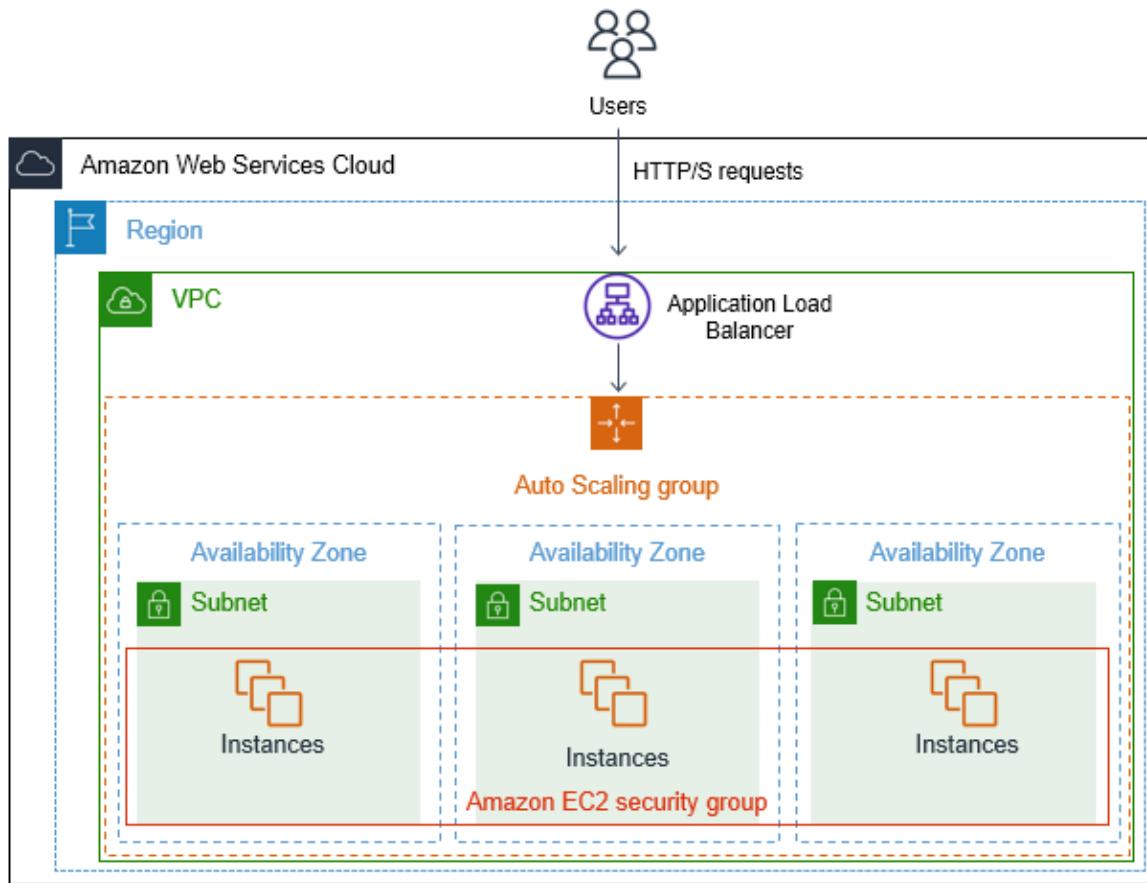
Tutorial: Set up a scaled and load-balanced application

Important

Before you explore this tutorial, we recommend that you first review the following introductory tutorial: [Get started with Amazon EC2 Auto Scaling \(p. 14\)](#).

Registering your Auto Scaling group with an Elastic Load Balancing load balancer helps you set up a load-balanced application. Elastic Load Balancing works with Amazon EC2 Auto Scaling to distribute incoming traffic across your healthy Amazon EC2 instances. This increases the scalability and availability of your application. You can enable Elastic Load Balancing within multiple Availability Zones to increase the fault tolerance of your applications.

In this tutorial, we cover the basics steps for setting up a load-balanced application when the Auto Scaling group is created. When complete, your architecture should look similar to the following diagram:



Elastic Load Balancing supports different types of load balancers. We recommend that you use an Application Load Balancer for this tutorial.

For more information about introducing a load balancer into your architecture, see [Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group \(p. 286\)](#).

Tasks

- [Prerequisites \(p. 298\)](#)
- [Step 1: Set up a launch template or launch configuration \(p. 299\)](#)
- [Step 2: Create an Auto Scaling group \(p. 301\)](#)
- [Step 3: Verify that your load balancer is attached \(p. 302\)](#)
- [Step 4: Next steps \(p. 302\)](#)
- [Step 5: Clean up \(p. 303\)](#)

Prerequisites

- A load balancer and target group. Make sure to choose the same Availability Zones for the load balancer that you plan to use for your Auto Scaling group. For more information, see [Getting started with Elastic Load Balancing](#) in the *Elastic Load Balancing User Guide*.
- A security group for your launch template or launch configuration. The security group must allow access from the load balancer on both the listener port (usually port 80 for HTTP traffic) and the port that you want Elastic Load Balancing to use for health checks. For more information, see the applicable documentation:

- [Target security groups](#) in the *User Guide for Application Load Balancers*
- [Target security groups](#) in the *User Guide for Network Load Balancers*

Optionally, if your instances will have public IP addresses, you can allow SSH traffic for connecting to the instances.

- (Optional) An IAM role that grants your application access to AWS.
- (Optional) An Amazon Machine Image (AMI) defined as the source template for your Amazon EC2 instances. To create one now, launch an instance. Specify the IAM role (if you created one) and any configuration scripts that you need as user data. Connect to the instance and customize it. For example, you can install software and applications, copy data, and attach additional EBS volumes. Test your applications on your instance to ensure that it is configured correctly. Save this updated configuration as a custom AMI. If you don't need the instance later, you can terminate it. Instances launched from this new custom AMI include the customizations that you made when you created the AMI.
- A virtual private cloud (VPC). This tutorial refers to the default VPC, but you can use your own. If using your own VPC, make sure that it has a subnet mapped to each Availability Zone of the Region you are working in. At minimum, you must have two public subnets available to create the load balancer. You must also have either two private subnets or two public subnets to create your Auto Scaling group and register it with the load balancer.

Step 1: Set up a launch template or launch configuration

Use either a launch template or a launch configuration for this tutorial.

If you already have a launch template that you'd like to use, select it by using the following procedure.

Note

Alternatively, you can use a launch configuration instead of a launch template. For the launch configuration instructions, see [Select or create a launch configuration \(p. 300\)](#).

To select an existing launch template

1. Open the [Launch templates page](#) of the Amazon EC2 console.
2. On the navigation bar at the top of the screen, choose the Region where the load balancer was created.
3. Select a launch template.
4. Choose **Actions, Create Auto Scaling group**.

Alternatively, to create a new launch template, use the following procedure.

To create a launch template

1. Open the [Launch templates page](#) of the Amazon EC2 console.
2. On the navigation bar at the top of the screen, choose the Region where the load balancer was created.
3. Choose **Create launch template**.
4. Enter a name and provide a description for the initial version of the launch template.
5. For **Application and OS Images (Amazon Machine Image)**, choose the ID of the AMI for your instances. You can search through all available AMIs, or select an AMI from the **Recents** or **Quick Start** list. If you don't see the AMI that you need, choose **Browse more AMIs** to browse the full AMI catalog.
6. For **Instance type**, select a hardware configuration for your instances that is compatible with the AMI that you specified.

7. (Optional) For **Key pair (login)**, choose the key pair to use when connecting to your instances.
8. For **Network settings**, expand **Advanced network configuration** and do the following:
 - a. Choose **Add network interface** to configure the primary network interface.
 - b. (Optional) For **Auto-assign public IP**, keep the default value, **Don't include in launch template**. When you create your Auto Scaling group, you can assign a public IPv4 address to instances in your Auto Scaling group by using subnets that have the public IPv4 addressing attribute enabled, such as the default subnets in the default VPC. Alternatively, if you don't need to connect to your instances, you can choose **Disable** to prevent instances in your group from receiving traffic directly from the internet. In this case, they will receive traffic only from the load balancer.
 - c. For **Security group ID**, specify a security group for your instances from the same VPC as the load balancer.
 - d. For **Delete on termination**, choose **Yes**. This deletes the network interface when the Auto Scaling group scales in, and terminates the instance to which the network interface is attached.
9. (Optional) To securely distribute credentials to your instances, for **Advanced details, IAM instance profile**, enter the Amazon Resource Name (ARN) of your IAM role.
10. (Optional) To specify user data or a configuration script for your instances, paste it into **Advanced details, User data**.
11. Choose **Create launch template**.
12. On the confirmation page, choose **Create Auto Scaling group**.

Select or create a launch configuration

If you already have a launch configuration that you'd like to use, select it by using the following procedure.

To select an existing launch configuration

1. Open the [Launch configurations page](#) of the Amazon EC2 console.
2. On the navigation bar at the top of the screen, choose the Region where the load balancer was created.
3. Select a launch configuration.
4. Choose **Actions, Create Auto Scaling group**.

Alternatively, to create a new launch configuration, use the following procedure.

To create a launch configuration

1. Open the [Launch configurations page](#) of the Amazon EC2 console.
2. On the navigation bar at the top of the screen, choose the Region where the load balancer was created.
3. Choose **Create launch configuration**, and enter a name for your launch configuration.
4. For **Amazon machine image (AMI)**, enter the ID of the AMI for your instances as search criteria.
5. For **Instance type**, select a hardware configuration for your instance.
6. Under **Additional configuration**, pay attention to the following fields:
 - a. (Optional) To securely distribute credentials to your EC2 instance, for **IAM instance profile**, select your IAM role. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).
 - b. (Optional) To specify user data or a configuration script for your instance, paste it into **Advanced details, User data**.

- c. (Optional) For **Advanced details, IP address type**, keep the default value. When you create your Auto Scaling group, you can assign a public IP address to instances in your Auto Scaling group by using subnets that have the public IP addressing attribute enabled, such as the default subnets in the default VPC. Alternatively, if you don't need to connect to your instances, you can choose **Do not assign a public IP address to any instances** to prevent instances in your group from receiving traffic directly from the internet. In this case, they will receive traffic only from the load balancer.
7. For **Security groups**, choose an existing security group from the same VPC as the load balancer. If you keep the **Create a new security group** option selected, a default SSH rule is configured for Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.
8. For **Key pair (login)**, choose an option under **Key pair options**.
If you've already configured an Amazon EC2 instance key pair, you can choose it here.
If you don't already have an Amazon EC2 instance key pair, choose **Create a new key pair** and give it a recognizable name. Choose **Download key pair** to download the key pair to your computer.
Important
If you need to connect to your instances, do not choose **Proceed without a key pair**.
9. Select the acknowledgment check box, and then choose **Create launch configuration**.
10. Select the check box next to the name of your new launch configuration and choose **Actions, Create Auto Scaling group**.

Step 2: Create an Auto Scaling group

Use the following procedure to continue where you left off after creating or selecting your launch template or launch configuration.

To create an Auto Scaling group

1. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.
2. [Launch template only] For **Launch template**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.
3. Choose **Next**.

The **Choose instance launch options** page appears, allowing you to choose the VPC network settings you want the Auto Scaling group to use and giving you options for launching On-Demand and Spot Instances (if you chose a launch template).

4. In the **Network** section, for **VPC**, choose the VPC that you used for your load balancer. If you chose the default VPC, it is automatically configured to provide internet connectivity to your instances. This VPC includes a public subnet in each Availability Zone in the Region.
5. For **Availability Zones and subnets**, choose one or more subnets from each Availability Zone that you want to include, based on which Availability Zones the load balancer is in. For more information, see [Considerations when choosing VPC subnets \(p. 319\)](#).
6. [Launch template only] In the **Instance type requirements** section, use the default setting to simplify this step. (Do not override the launch template.) For this tutorial, you will launch only On-Demand Instances using the instance type specified in your launch template.
7. Choose **Next** to go to the **Configure advanced options** page.
8. To attach the group to an existing load balancer, in the **Load balancing** section, choose **Attach to an existing load balancer**. You can choose **Choose from your load balancer target groups** or **Choose from Classic Load Balancers**. You can then choose the name of a target group for the Application Load Balancer or Network Load Balancer you created, or choose the name of a Classic Load Balancer.

9. (Optional) To use Elastic Load Balancing health checks, for **Health checks**, choose **ELB** under **Health check type**.
10. When you have finished configuring the Auto Scaling group, choose **Skip to review**.
11. On the **Review** page, review the details of your Auto Scaling group. You can choose **Edit** to make changes. When you are finished, choose **Create Auto Scaling group**.

After you have created the Auto Scaling group with the load balancer attached, the load balancer automatically registers new instances as they come online. You have only one instance at this point, so there isn't much to register. However, you can add additional instances by updating the desired capacity of the group. For step-by-step instructions, see [Manual scaling \(p. 124\)](#).

Step 3: Verify that your load balancer is attached

To verify that your load balancer is attached

1. From the [Auto Scaling groups page](#) of the Amazon EC2 console, select the check box next to your Auto Scaling group.
2. On the **Details** tab, **Load balancing** shows any attached load balancer target groups or Classic Load Balancers.
3. On the **Activity** tab, in **Activity history**, you can verify that your instances launched successfully. The **Status** column shows whether your Auto Scaling group has successfully launched instances. If your instances fail to launch, you can find troubleshooting ideas for common instance launch issues in [Troubleshoot Amazon EC2 Auto Scaling \(p. 360\)](#).
4. On the **Instance management** tab, under **Instances**, you can verify that your instances are ready to receive traffic. Initially, your instances are in the **Pending** state. After an instance is ready to receive traffic, its state is **InService**. The **Health status** column shows the result of the Amazon EC2 Auto Scaling health checks on your instances. Although an instance may be marked as healthy, the load balancer will only send traffic to instances that pass the load balancer health checks.
5. Verify that your instances are registered with the load balancer. Open the [Target groups page](#) of the Amazon EC2 console. Select your target group, and then choose the **Targets** tab. If the state of your instances is **initial**, it's probably because they are still in the process of being registered, or they are still undergoing health checks. When the state of your instances is **healthy**, they are ready for use.

Step 4: Next steps

Now that you have completed this tutorial, you can learn more:

- You can configure your Auto Scaling group to use Elastic Load Balancing health checks. If you enable load balancer health checks and an instance fails the health checks, the Auto Scaling group considers the instance unhealthy and replaces it. For more information, see [Add Elastic Load Balancing health checks \(p. 291\)](#).
- You can expand your application to an additional Availability Zone in the same Region to increase fault tolerance if there is a service disruption. For more information, see [Add Availability Zones \(p. 292\)](#).
- You can configure your Auto Scaling group to use a target tracking scaling policy. This automatically increases or decreases the number of instances as the demand on your instances changes. This allows the group to handle changes in the amount of traffic that your application receives. For more information, see [Target tracking scaling policies \(p. 135\)](#).

Step 5: Clean up

After you're finished with the resources that you created for this tutorial, you should consider cleaning them up to avoid incurring unnecessary charges.

To delete your Auto Scaling group

1. Open the [Auto Scaling groups page](#) of the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.
3. Choose **Delete**.
4. When prompted for confirmation, type **delete** to confirm deleting the specified Auto Scaling group and then choose **Delete**.

A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. When the deletion has occurred, the **Desired**, **Min**, and **Max** columns show **0** instances for the Auto Scaling group. It takes a few minutes to terminate the instance and delete the group. Refresh the list to see the current state.

Skip the following procedure if you would like to keep your launch template.

To delete your launch template

1. Open the [Launch templates page](#) of the Amazon EC2 console.
2. Select your launch template.
3. Choose **Actions, Delete template**.
4. When prompted for confirmation, type **Delete** to confirm deleting the specified launch template and then choose **Delete**.

Skip the following procedure if you would like to keep your launch configuration.

To delete your launch configuration

1. Open the [Launch configurations page](#) of the Amazon EC2 console.
2. Select your launch configuration.
3. Choose **Actions, Delete launch configuration**.
4. When prompted for confirmation, choose **Delete**.

Skip the following procedure if you want to keep the load balancer for future use.

To delete your load balancer

1. Open the [Load balancers page](#) of the Amazon EC2 console.
2. Choose the load balancer and choose **Actions, Delete**.
3. When prompted for confirmation, choose **Yes, Delete**.

To delete your target group

1. Open the [Target groups page](#) of the Amazon EC2 console.
2. Choose the target group and choose **Actions, Delete**.
3. When prompted for confirmation, choose **Yes, Delete**.

Use EventBridge to handle Auto Scaling events

Amazon EventBridge, formerly called CloudWatch Events, helps you set up event-driven rules that monitor resources and initiate target actions that use other AWS services.

Events from Amazon EC2 Auto Scaling are delivered to EventBridge in near real time. You can establish EventBridge rules that invoke programmatic actions and notifications in response to a variety of these events. For example, while instances are in the process of launching or terminating, you can invoke an AWS Lambda function to perform a preconfigured task.

Targets of EventBridge rules can include AWS Lambda functions, Amazon SNS topics, API destinations, event buses in other AWS accounts, and many more. For information about supported targets, see [Amazon EventBridge targets](#) in the *Amazon EventBridge User Guide*.

Get started by creating EventBridge rules with an example using an Amazon SNS topic and an EventBridge rule. Then, when a user starts an instance refresh, Amazon SNS notifies you by email whenever a checkpoint is reached. For more information, see [Create EventBridge rules for instance refresh events \(p. 314\)](#).

Contents

- [Amazon EC2 Auto Scaling event reference \(p. 304\)](#)
- [Warm pool event types and patterns \(p. 309\)](#)
- [Create EventBridge rules \(p. 314\)](#)

Amazon EC2 Auto Scaling event reference

Using Amazon EventBridge, you can create *rules* that match incoming *events* and route them to *targets* for processing.

Example events

The following examples show events for Amazon EC2 Auto Scaling. Events are produced on a best-effort basis.

Event types

- [EC2 Instance-launch Lifecycle Action \(p. 304\)](#)
- [EC2 Instance Launch Successful \(p. 305\)](#)
- [EC2 Instance Launch Unsuccessful \(p. 306\)](#)
- [EC2 Instance-terminate Lifecycle Action \(p. 306\)](#)
- [EC2 Instance Terminate Successful \(p. 307\)](#)
- [EC2 Instance Terminate Unsuccessful \(p. 307\)](#)
- [EC2 Auto Scaling Instance Refresh Checkpoint Reached \(p. 308\)](#)
- [EC2 Auto Scaling Instance Refresh Started \(p. 308\)](#)
- [EC2 Auto Scaling Instance Refresh Succeeded \(p. 308\)](#)
- [EC2 Auto Scaling Instance Refresh Failed \(p. 309\)](#)
- [EC2 Auto Scaling Instance Refresh Cancelled \(p. 309\)](#)

EC2 Instance-launch Lifecycle Action

In this example event, Amazon EC2 Auto Scaling moved an instance to a Pending:Wait state due to a lifecycle hook.

Note

Amazon EC2 Auto Scaling supports providing Origin and Destination in events for lifecycle hooks, if that information is available. For more information, see [Warm pool event types and patterns \(p. 309\)](#).

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance-launch Lifecycle Action",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "yyyy-mm-ddThh:mm:ssZ",  
  "region": "us-west-2",  
  "resources": [  
    "auto-scaling-group-arn"  
  ],  
  "detail": {  
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",  
    "AutoScalingGroupName": "my-asg",  
    "LifecycleHookName": "my-lifecycle-hook",  
    "EC2InstanceId": "i-1234567890abcdef0",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",  
    "NotificationMetadata": "additional-info"  
  }  
}
```

EC2 Instance Launch Successful

In this example event, Amazon EC2 Auto Scaling successfully launched an instance.

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance Launch Successful",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "yyyy-mm-ddThh:mm:ssZ",  
  "region": "us-west-2",  
  "resources": [  
    "auto-scaling-group-arn",  
    "instance-arn"  
  ],  
  "detail": {  
    "StatusCode": "InProgress",  
    "Description": "Launching a new EC2 instance: i-12345678",  
    "AutoScalingGroupName": "my-auto-scaling-group",  
    "ActivityId": "87654321-4321-4321-4321-210987654321",  
    "Details": {  
      "Availability Zone": "us-west-2b",  
      "Subnet ID": "subnet-12345678"  
    },  
    "RequestId": "12345678-1234-1234-1234-123456789012",  
    "StatusMessage": "",  
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",  
    "EC2InstanceId": "i-1234567890abcdef0",  
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",  
    "Cause": "description-text"  
  }  
}
```

EC2 Instance Launch Unsuccessful

In this example event, Amazon EC2 Auto Scaling failed to launch an instance.

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance Launch Unsuccessful",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "yyyy-mm-ddThh:mm:ssZ",  
  "region": "us-west-2",  
  "resources": [  
    "auto-scaling-group-arn",  
    "instance-arn"  
,  
  "detail": {  
    "StatusCode": "Failed",  
    "AutoScalingGroupName": "my-auto-scaling-group",  
    "ActivityId": "87654321-4321-4321-4321-210987654321",  
    "Details": {  
      "Availability Zone": "us-west-2b",  
      "Subnet ID": "subnet-12345678"  
    },  
    "RequestId": "12345678-1234-1234-1234-123456789012",  
    "StatusMessage": "message-text",  
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",  
    "EC2InstanceId": "i-1234567890abcdef0",  
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",  
    "Cause": "description-text"  
  }  
}
```

EC2 Instance-terminate Lifecycle Action

In this example event, Amazon EC2 Auto Scaling moved an instance to a Terminating:Wait state due to a lifecycle hook.

Note

Amazon EC2 Auto Scaling supports providing Origin and Destination in events for lifecycle hooks, if that information is available. For more information, see [Warm pool event types and patterns \(p. 309\)](#).

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance-terminate Lifecycle Action",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "yyyy-mm-ddThh:mm:ssZ",  
  "region": "us-west-2",  
  "resources": [  
    "auto-scaling-group-arn"  
,  
  "detail": {  
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",  
    "AutoScalingGroupName": "my-asg",  
    "LifecycleHookName": "my-lifecycle-hook",  
    "EC2InstanceId": "i-1234567890abcdef0",  
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",  
    "NotificationMetadata": "additional-info"  
  }  
}
```

}

EC2 Instance Terminate Successful

In this example event, Amazon EC2 Auto Scaling successfully terminated an instance.

```
{  
    "version": "0",  
    "id": "12345678-1234-1234-1234-123456789012",  
    "detail-type": "EC2 Instance Terminate Successful",  
    "source": "aws.autoscaling",  
    "account": "123456789012",  
    "time": "yyyy-mm-ddThh:mm:ssZ",  
    "region": "us-west-2",  
    "resources": [  
        "auto-scaling-group-arn",  
        "instance-arn"  
    ],  
    "detail": {  
        "StatusCode": "InProgress",  
        "Description": "Terminating EC2 instance: i-12345678",  
        "AutoScalingGroupName": "my-auto-scaling-group",  
        "ActivityId": "87654321-4321-4321-4321-210987654321",  
        "Details": {  
            "Availability Zone": "us-west-2b",  
            "Subnet ID": "subnet-12345678"  
        },  
        "RequestId": "12345678-1234-1234-1234-123456789012",  
        "StatusMessage": "",  
        "EndTime": "yyyy-mm-ddThh:mm:ssZ",  
        "EC2InstanceId": "i-1234567890abcdef0",  
        "StartTime": "yyyy-mm-ddThh:mm:ssZ",  
        "Cause": "description-text"  
    }  
}
```

EC2 Instance Terminate Unsuccessful

In this example event, Amazon EC2 Auto Scaling failed to terminate an instance.

```
{  
    "version": "0",  
    "id": "12345678-1234-1234-1234-123456789012",  
    "detail-type": "EC2 Instance Terminate Unsuccessful",  
    "source": "aws.autoscaling",  
    "account": "123456789012",  
    "time": "yyyy-mm-ddThh:mm:ssZ",  
    "region": "us-west-2",  
    "resources": [  
        "auto-scaling-group-arn",  
        "instance-arn"  
    ],  
    "detail": {  
        "StatusCode": "Failed",  
        "AutoScalingGroupName": "my-auto-scaling-group",  
        "ActivityId": "87654321-4321-4321-4321-210987654321",  
        "Details": {  
            "Availability Zone": "us-west-2b",  
            "Subnet ID": "subnet-12345678"  
        },  
        "RequestId": "12345678-1234-1234-1234-123456789012",  
        "StatusMessage": "message-text",  
        "Cause": "description-text"  
    }  
}
```

```

        "EndTime": "yyyy-mm-ddThh:mm:ssZ",
        "EC2InstanceId": "i-1234567890abcdef0",
        "StartTime": "yyyy-mm-ddThh:mm:ssZ",
        "Cause": "description-text"
    }
}

```

EC2 Auto Scaling Instance Refresh Checkpoint Reached

In this example event, the number of instances that have been replaced reaches the percentage threshold defined for the checkpoint.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
        "auto-scaling-group-arn"
    ],
    "detail": {
        "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
        "AutoScalingGroupName": "my-auto-scaling-group",
        "CheckpointPercentage": "50",
        "CheckpointDelay": "300"
    }
}

```

EC2 Auto Scaling Instance Refresh Started

In this example event, the status of an instance refresh changes to InProgress.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Started",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
        "auto-scaling-group-arn"
    ],
    "detail": {
        "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
        "AutoScalingGroupName": "my-auto-scaling-group"
    }
}

```

EC2 Auto Scaling Instance Refresh Succeeded

In this example event, the status of an instance refresh changes to Succeeded.

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
}

```

```

"source": "aws.autoscaling",
"account": "123456789012",
"time": "yyyy-mm-ddThh:mm:ssZ",
"region": "us-west-2",
"resources": [
    "auto-scaling-group-arn"
],
"detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-auto-scaling-group"
}
}

```

EC2 Auto Scaling Instance Refresh Failed

In this example event, the status of an instance refresh changes to Failed.

```

{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Failed",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
        "auto-scaling-group-arn"
    ],
    "detail": {
        "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
        "AutoScalingGroupName": "my-auto-scaling-group"
    }
}

```

EC2 Auto Scaling Instance Refresh Cancelled

In this example event, the status of an instance refresh changes to Cancelled.

```

{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
        "auto-scaling-group-arn"
    ],
    "detail": {
        "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
        "AutoScalingGroupName": "my-auto-scaling-group"
    }
}

```

Warm pool event types and patterns

Note

Amazon EC2 Auto Scaling supports several predefined patterns in Amazon EventBridge. This simplifies how an event pattern is created. You select field values on a form, and EventBridge

generates the pattern for you. At this time, Amazon EC2 Auto Scaling doesn't support predefined patterns for any events that are emitted by an Auto Scaling group with a warm pool. You must enter the pattern as a JSON object. This section and the [Create EventBridge rules for warm pool events \(p. 316\)](#) topic show you how to use an event pattern to select events and send them to targets.

When you add lifecycle hooks to your Auto Scaling group, events are sent to EventBridge when an instance transitions into a wait state.

There are two primary event types for lifecycle hooks:

- EC2 Instance-launch Lifecycle Action
- EC2 Instance-terminate Lifecycle Action

To create EventBridge rules that filter for specific events that are emitted by an Auto Scaling group with a warm pool, include the Origin and Destination fields from the detail section of the event. Events that contain the Origin and Destination fields are only emitted by Auto Scaling groups that have a warm pool.

The values of Origin and Destination can be the following:

EC2 | AutoScalingGroup | WarmPool

Contents

- [Example events \(p. 310\)](#)
- [Example event patterns \(p. 312\)](#)

Example events

This section lists example events that are sent to EventBridge when an instance transitions into a wait state. Events are emitted on a best-effort basis.

Event examples

- [Example 1: Amazon EC2 Auto Scaling adds a new instance to the warm pool \(p. 310\)](#)
- [Example 2: Amazon EC2 Auto Scaling adds an instance to the Auto Scaling group \(p. 311\)](#)
- [Example 3: Amazon EC2 Auto Scaling adds a new instance to the Auto Scaling group \(p. 311\)](#)
- [Example 4: Amazon EC2 Auto Scaling returns an instance to the warm pool \(p. 312\)](#)

Example 1: Amazon EC2 Auto Scaling adds a new instance to the warm pool

In this example event, the state of a new instance changes to Warmed:Pending:Wait when it is added to a warm pool. This occurs because of a lifecycle hook for scale-out events.

```
{  
  "version": "0",  
  "id": "12345678-1234-1234-1234-123456789012",  
  "detail-type": "EC2 Instance-launch Lifecycle Action",  
  "source": "aws.autoscaling",  
  "account": "123456789012",  
  "time": "2021-01-13T00:12:37.214Z",  
  "region": "us-west-2",  
  "resources": [  
    "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
```

```

        ],
        "detail": {
            "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-lifecycle-hook",
            "EC2InstanceId": "i-1234567890abcdef0",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
            "NotificationMetadata": "additional-info",
            "Origin": "EC2",
            "Destination": "WarmPool"
        }
    }
}

```

Example 2: Amazon EC2 Auto Scaling adds an instance to the Auto Scaling group

In this example event, the state of an instance in the warm pool changes to `Pending:Wait` when it is added to the Auto Scaling group. This occurs because of a lifecycle hook for scale-out events.

```

{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Instance-launch Lifecycle Action",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "2021-01-19T00:35:52.359Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
    ],
    "detail": {
        "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
        "AutoScalingGroupName": "my-asg",
        "LifecycleHookName": "my-launch-lifecycle-hook",
        "EC2InstanceId": "i-1234567890abcdef0",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationMetadata": "additional-info",
        "Origin": "WarmPool",
        "Destination": "AutoScalingGroup"
    }
}

```

Example 3: Amazon EC2 Auto Scaling adds a new instance to the Auto Scaling group

In this example event, the state of a new instance (not an instance from the warm pool) changes to `Pending:Wait` when it is added to the Auto Scaling group. This occurs because of a lifecycle hook for scale-out events.

```

{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Instance-launch Lifecycle Action",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "2021-02-01T17:18:06.082Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
    ]
}

```

```

        ],
        "detail": {
            "LifecycleActionToken": "87654321-4321-4321-4321-21098EXAMPLE",
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-lifecycle-hook",
            "EC2InstanceId": "i-1234567890abcdef0",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
            "NotificationMetadata": "additional-info",
            "Origin": "EC2",
            "Destination": "AutoScalingGroup"
        }
    }
}

```

Example 4: Amazon EC2 Auto Scaling returns an instance to the warm pool

In this example event, the state of an instance changes to `Warmed:Pending:Wait` when it is returned to the warm pool. This occurs because of a lifecycle hook for scale-in events.

```

{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "EC2 Instance-terminate Lifecycle Action",
    "source": "aws.autoscaling",
    "account": "123456789012",
    "time": "2022-03-28T00:12:37.214Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
    ],
    "detail": {
        "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
        "AutoScalingGroupName": "my-asg",
        "LifecycleHookName": "my-termination-lifecycle-hook",
        "EC2InstanceId": "i-1234567890abcdef0",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "NotificationMetadata": "additional-info",
        "Origin": "AutoScalingGroup",
        "Destination": "WarmPool"
    }
}

```

Example event patterns

The preceding section provides example events emitted by Amazon EC2 Auto Scaling.

EventBridge event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

The following fields in the event form the event pattern that is defined in the rule to invoke an action:

`"source": "aws.autoscaling"`

Identifies that the event is from Amazon EC2 Auto Scaling.

`"detail-type": "EC2 Instance-launch Lifecycle Action"`

Identifies the event type.

`"Origin": "EC2"`

Identifies where the instance is coming from.

"Destination": "WarmPool"

Identifies where the instance is going to.

Use the following sample event pattern to capture all events that are associated with instances entering the warm pool.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],  
  "detail": {  
    "Origin": [ "EC2" ],  
    "Destination": [ "WarmPool" ]  
  }  
}
```

Use the following sample event pattern to capture all events that are associated with instances leaving the warm pool because of a scale-out event.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],  
  "detail": {  
    "Origin": [ "WarmPool" ],  
    "Destination": [ "AutoScalingGroup" ]  
  }  
}
```

Use the following sample event pattern to capture all events that are associated with instances launching directly into the Auto Scaling group.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],  
  "detail": {  
    "Origin": [ "EC2" ],  
    "Destination": [ "AutoScalingGroup" ]  
  }  
}
```

Use the following sample event pattern to capture all events that are associated with instances returning to the warm pool on scale in.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],  
  "detail": {  
    "Origin": [ "AutoScalingGroup" ],  
    "Destination": [ "WarmPool" ]  
  }  
}
```

Use the following sample event pattern to capture all events that are associated with EC2 Instance-launch Lifecycle Action, regardless of the origin or destination.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
```

}

Create EventBridge rules

When an event is emitted by Amazon EC2 Auto Scaling, an event notification is sent to Amazon EventBridge as a JSON file. You can write an EventBridge rule to automate what actions to take when an event pattern matches the rule. If EventBridge detects an event pattern that matches a pattern defined in a rule, EventBridge invokes the target (or targets) specified in the rule.

You can use the example procedures in this section as a starting point.

You may also find the following documentation useful.

- To perform custom actions on instances as they are launching or before they are terminated using a Lambda function, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#).
- To invoke a Lambda function on API calls logged with CloudTrail, see [Tutorial: Log AWS API calls using EventBridge](#) in the *Amazon EventBridge User Guide*.
- For more information about how to create event rules, see [Creating Amazon EventBridge rules that react to events](#) in the *Amazon EventBridge User Guide*.

Topics

- [Create EventBridge rules for instance refresh events \(p. 314\)](#)
- [Create EventBridge rules for warm pool events \(p. 316\)](#)

Create EventBridge rules for instance refresh events

The following example creates an EventBridge rule to send an email notification. It does this each time that your Auto Scaling group emits an event when a checkpoint is reached during an instance refresh. The procedure for setting up email notifications using Amazon SNS is included. To use Amazon SNS to send email notifications, you must first create a *topic* and then subscribe your email addresses to the topic.

For more information about the instance refresh feature, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Create an Amazon SNS topic

An SNS topic is a logical access point, a communication channel that your Auto Scaling group uses to send the notifications. You create a topic by specifying a name for your topic.

Topic names must meet the following requirements:

- Have 1-256 characters
- Contain uppercase and lowercase ASCII letters, numbers, underscores, or hyphens

For more information, see [Creating an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Subscribe to the Amazon SNS topic

To receive the notifications that your Auto Scaling group sends to the topic, you must subscribe an endpoint to the topic. In this procedure, for **Endpoint**, specify the email address where you want to receive the notifications from Amazon EC2 Auto Scaling.

For more information, see [Subscribing to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Confirm your Amazon SNS subscription

Amazon SNS sends a confirmation email to the email address you specified in the previous step.

Make sure that you open the email from AWS Notifications and choose the link to confirm the subscription before you continue with the next step.

You will receive an acknowledgment message from AWS. Amazon SNS is now configured to receive notifications and send the notification as an email to the email address that you specified.

Route events to your Amazon SNS topic

Create a rule that matches selected events and routes them to your Amazon SNS topic to notify subscribed email addresses.

To create a rule that sends notifications to your Amazon SNS topic

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. For **Define rule detail**, do the following:
 - a. Enter a **Name** for the rule, and, optionally, a description.
A rule can't have the same name as another rule in the same Region and on the same event bus.
 - b. For **Event bus**, choose **default**. When an AWS service in your account generates an event, it always goes to your account's default event bus.
 - c. For **Rule type**, choose **Rule with an event pattern**.
 - d. Choose **Next**.
5. For **Build event pattern**, do the following:
 - a. For **Event source**, choose **AWS events or EventBridge partner events**.
 - b. For **Event pattern**, do the following:
 - i. For **Event source**, choose **AWS services**.
 - ii. For **AWS service**, choose **Auto Scaling**.
 - iii. For **Event type**, choose **Instance Refresh**.
 - iv. By default, the rule matches any instance refresh event. To create a rule that notifies you when a checkpoint is reached during an instance refresh, choose **Specific instance event(s)** and select **EC2 Auto Scaling Instance Refresh Checkpoint Reached**.
 - v. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific Auto Scaling group, choose **Specific group name(s)** and select one or more Auto Scaling groups.
 - vi. Choose **Next**.
6. For **Select target(s)**, do the following:
 - a. For **Target types**, choose **AWS service**.
 - b. For **Select a target**, choose **SNS topic**.
 - c. For **Topic**, choose your Amazon SNS topic.
 - d. (Optional) Under **Additional settings**, you can optionally configure additional settings. For more information, see [Creating Amazon EventBridge rules that react to events](#) (step 16) in the *Amazon EventBridge User Guide*.

- e. Choose **Next**.
7. (Optional) For **Tags**, you can optionally assign one or more tags to your rule, and then choose **Next**.
8. For **Review and create**, review the details of the rule and modify them as necessary. Then, choose **Create rule**.

Create EventBridge rules for warm pool events

The following example creates an EventBridge rule to invoke programmatic actions. It does this each time that your Auto Scaling group emits an event when a new instance is added to the warm pool.

Before you create the rule, create the AWS Lambda function that you want the rule to use as a target. You must specify this function as the target for the rule. The following procedure provides only the steps for creating the EventBridge rule that acts when new instances enter the warm pool. For an introductory tutorial that shows you how to create a simple Lambda function to invoke when an incoming event matches a rule, see [Tutorial: Configure a lifecycle hook that invokes a Lambda function \(p. 213\)](#).

For more information about creating and working with warm pools, see [Warm pools for Amazon EC2 Auto Scaling \(p. 219\)](#).

To create an event rule that invokes a Lambda function

1. Open the Amazon EventBridge console at <https://console.aws.amazon.com/events/>.
2. In the navigation pane, choose **Rules**.
3. Choose **Create rule**.
4. For **Define rule detail**, do the following:
 - a. Enter a **Name** for the rule, and, optionally, a description.
A rule can't have the same name as another rule in the same Region and on the same event bus.
 - b. For **Event bus**, choose **default**. When an AWS service in your account generates an event, it always goes to your account's default event bus.
 - c. For **Rule type**, choose **Rule with an event pattern**.
 - d. Choose **Next**.
5. For **Build event pattern**, do the following:
 - a. For **Event source**, choose **AWS events or EventBridge partner events**.
 - b. For **Event pattern**, choose **Custom pattern (JSON editor)**, and paste the following pattern into the **Event pattern** box, replacing the text in *italics* with the name of your Auto Scaling group.

```
{  
  "source": [ "aws.autoscaling" ],  
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],  
  "detail": {  
    "AutoScalingGroupName": [ "my-asg" ],  
    "Origin": [ "EC2" ],  
    "Destination": [ "WarmPool" ]  
  }  
}
```

To create a rule that matches for other events, modify the event pattern. For more information, see [Example event patterns \(p. 312\)](#).

- c. Choose **Next**.
6. For **Select target(s)**, do the following:

- a. For **Target types**, choose **AWS service**.
 - b. For **Select a target**, choose **Lambda function**.
 - c. For **Function**, choose the function that you want to send the events to.
 - d. (Optional) For **Configure version/alias**, enter version and alias settings for the target Lambda function.
 - e. (Optional) For **Additional settings**, enter any additional settings as appropriate for your application. For more information, see [Creating Amazon EventBridge rules that react to events \(step 16\)](#) in the *Amazon EventBridge User Guide*.
 - f. Choose **Next**.
7. (Optional) For **Tags**, you can optionally assign one or more tags to your rule, and then choose **Next**.
 8. For **Review and create**, review the details of the rule and modify them as necessary. Then, choose **Create rule**.

Provide network connectivity for your Auto Scaling instances using Amazon VPC

We are retiring EC2-Classic. We recommend that you migrate from EC2-Classic to a VPC. For more information, see the blog post [EC2-Classic Networking is Retiring - Here's How to Prepare](#).

Amazon Virtual Private Cloud (Amazon VPC) enables you to define a virtual networking environment in a private, isolated section of the AWS Cloud. You have complete control over your virtual networking environment.

Within a virtual private cloud (VPC), you can launch AWS resources such as Auto Scaling groups. An Auto Scaling group in a VPC works essentially the same way as it does on EC2-Classic and supports the same set of features.

A subnet in Amazon VPC is a subdivision within an Availability Zone defined by a segment of the IP address range of the VPC. Using subnets, you can group your instances based on your security and operational needs. A subnet resides entirely within the Availability Zone it was created in. You launch Auto Scaling instances within the subnets.

To enable communication between the internet and the instances in your subnets, you must create an internet gateway and attach it to your VPC. An internet gateway enables your resources within the subnets to connect to the internet through the Amazon EC2 network edge. If a subnet's traffic is routed to an internet gateway, the subnet is known as a *public* subnet. If a subnet's traffic is not routed to an internet gateway, the subnet is known as a *private* subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that need not be connected to the internet. For more information about giving internet access to instances in a VPC, see [Accessing the internet](#) in the *Amazon VPC User Guide*.

Contents

- [EC2-Classic \(p. 318\)](#)
- [Default VPC \(p. 318\)](#)
- [Nondefault VPC \(p. 319\)](#)
- [Considerations when choosing VPC subnets \(p. 319\)](#)
- [IP addressing in a VPC \(p. 319\)](#)

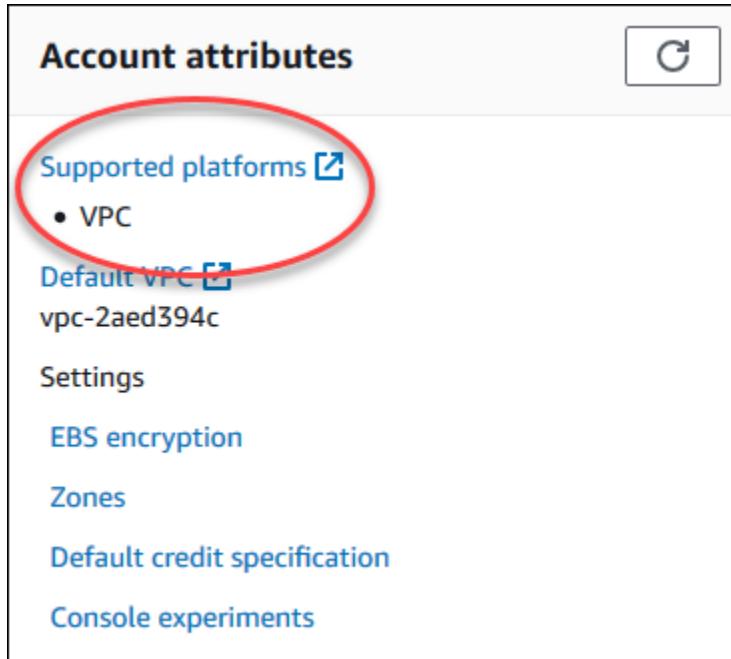
- [Network interfaces in a VPC \(p. 320\)](#)
- [Instance placement tenancy \(p. 320\)](#)
- [AWS Outposts \(p. 320\)](#)
- [More resources for learning about VPCs \(p. 320\)](#)

EC2-Classic

If you created your AWS account before December 4, 2013, it may allow you to choose between Amazon VPC and EC2-Classic in certain Regions. If you have one of these older accounts, you might have Auto Scaling groups in EC2-Classic in some Regions instead of Amazon VPC.

For information about migrating from EC2-Classic to a VPC, see the blog post [EC2-Classic Networking is Retiring - Here's How to Prepare](#). For information about the differences between instances in EC2-Classic and a VPC, see [EC2-Classic](#) in the *Amazon EC2 User Guide for Linux Instances*.

To determine whether any of the AWS Regions you use are still using EC2-Classic, open the Amazon EC2 console. If **Supported platforms** indicates only VPC, as shown in the following example, your AWS account in the current AWS Region uses the VPC platform, and uses a default VPC. The name of the default VPC is shown below the supported platform.



Tip

Any Auto Scaling groups created in a Region that has a default VPC can launch instances into the default VPC or a nondefault VPC, but not in EC2-Classic.

Default VPC

If you created your AWS account after December 4, 2013 or you are creating your Auto Scaling group in a new AWS Region, we create a default VPC for you. Your default VPC comes with a default subnet in each Availability Zone. If you have a default VPC, your Auto Scaling group is created in the default VPC by default.

You can view your VPCs on the [Your VPCs page](#) of the Amazon VPC console.

For more information about the default VPC, see [Default VPC and default subnets in the Amazon VPC User Guide](#).

Nondefault VPC

You can choose to create additional VPCs by going to the [VPC Dashboard page](#) in the AWS Management Console and selecting **Create VPC**.

For more information, see the [Amazon VPC User Guide](#).

Note

A VPC spans all Availability Zones in its AWS Region. When you add subnets to your VPC, choose multiple Availability Zones to ensure that the applications hosted in those subnets are highly available. An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Availability Zones help you to make production applications highly available, fault tolerant, and scalable.

Considerations when choosing VPC subnets

Note the following considerations when choosing VPC subnets for your Auto Scaling group:

- If you're attaching an Elastic Load Balancing load balancer to your Auto Scaling group, the instances can be launched into either public or private subnets. However, the load balancer can be created in public subnets only.
- If you're accessing your Auto Scaling instances directly through SSH, the instances can be launched into public subnets only.
- If you're accessing no-ingress Auto Scaling instances using AWS Systems Manager Session Manager, the instances can be launched into either public or private subnets.
- If you're using private subnets, you can allow the Auto Scaling instances to access the internet by using a public NAT gateway.
- By default, the default subnets in a default VPC are public subnets.

IP addressing in a VPC

When you launch your Auto Scaling instances in a VPC, your instances are automatically assigned a private IP address from the CIDR range of the subnet in which the instance is launched. This enables your instances to communicate with other instances in the VPC.

You can configure a launch template or launch configuration to assign public IPv4 addresses to your instances. Assigning public IP addresses to your instances enables them to communicate with the internet or other AWS services.

When you launch instances into a subnet that is configured to automatically assign IPv6 addresses, they receive both IPv4 and IPv6 addresses. Otherwise, they receive only IPv4 addresses. For more information, see [IPv6 addresses](#) in the *Amazon EC2 User Guide for Linux Instances*.

For information on specifying CIDR ranges for your VPC or subnet, see the [Amazon VPC User Guide](#).

Amazon EC2 Auto Scaling can automatically assign additional private IP addresses on instance launch when you use a launch template that specifies additional network interfaces. Each network interface is assigned a single private IP address from the CIDR range of the subnet in which the instance is launched. In this case, the system can no longer auto-assign a public IPv4 address to the primary network interface. You will not be able to connect to your instances over a public IPv4 address unless you associate available Elastic IP addresses to the Auto Scaling instances.

Network interfaces in a VPC

Each instance in your VPC has a default network interface (the primary network interface). You cannot detach a primary network interface from an instance. You can create and attach an additional network interface to any instance in your VPC. The number of network interfaces you can attach varies by instance type.

When launching an instance using a launch template, you can specify additional network interfaces. However, launching an Auto Scaling instance with multiple network interfaces automatically creates each interface in the same subnet as the instance. This is because Amazon EC2 Auto Scaling ignores the subnets defined in the launch template in favor of what is specified in the Auto Scaling group. For more information, see [Creating a launch template for an Auto Scaling group](#).

If you create or attach two or more network interfaces from the same subnet to an instance, you might encounter networking issues such as asymmetric routing, especially on instances using a variant of non-Amazon Linux. If you need this type of configuration, you must configure the secondary network interface within the OS. For an example, see [How can I make my secondary network interface work in my Ubuntu EC2 instance?](#) in the AWS Knowledge Center.

Instance placement tenancy

By default, all instances in the VPC run as shared tenancy instances. Amazon EC2 Auto Scaling also supports Dedicated Instances and Dedicated Hosts. However, support for Dedicated Hosts is only available for Auto Scaling groups that use a launch template. For more information, see [Configure instance tenancy with a launch configuration \(p. 48\)](#).

AWS Outposts

AWS Outposts extends an Amazon VPC from an AWS Region to an Outpost with the VPC components that are accessible in the Region, including internet gateways, virtual private gateways, Amazon VPC Transit Gateways, and VPC endpoints. An Outpost is homed to an Availability Zone in the Region and is an extension of that Availability Zone that you can use for resiliency.

For more information, see the [AWS Outposts User Guide](#).

For an example of how to deploy an Auto Scaling group that serves traffic from an Application Load Balancer within an Outpost, see the following blog post [Configuring an Application Load Balancer on AWS Outposts](#).

More resources for learning about VPCs

Use the following topics to learn more about VPCs and subnets.

- Private subnets in a VPC
 - [VPC with public and private subnets \(NAT\)](#)
 - [NAT gateways](#)
- Public subnets in a VPC
 - [VPC with a single public subnet](#)
- Subnets for your Application Load Balancer
 - [Subnets for your load balancer](#)
- General VPC information
 - [Amazon VPC User Guide](#)
 - [VPC peering](#)

- [Elastic network interfaces](#)
- [Use VPC endpoints for private connectivity \(p. 358\)](#)
- [Migrate from EC2-Classic to a VPC](#)

Security in Amazon EC2 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 Amazon EC2 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 Amazon EC2 Auto Scaling. The following topics show you how to configure Amazon EC2 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 Amazon EC2 Auto Scaling resources.

Topics

- [Amazon EC2 Auto Scaling and data protection \(p. 322\)](#)
- [Identity and Access Management for Amazon EC2 Auto Scaling \(p. 323\)](#)
- [Compliance validation for Amazon EC2 Auto Scaling \(p. 357\)](#)
- [Resilience in Amazon EC2 Auto Scaling \(p. 358\)](#)
- [Infrastructure security in Amazon EC2 Auto Scaling \(p. 358\)](#)
- [Amazon EC2 Auto Scaling and interface VPC endpoints \(p. 358\)](#)

Amazon EC2 Auto Scaling and data protection

The AWS [shared responsibility model](#) applies to data protection in Amazon EC2 Auto Scaling. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.

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

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Amazon EC2 Auto Scaling or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Use AWS KMS keys to encrypt Amazon EBS volumes

You can configure your Auto Scaling group to encrypt Amazon EBS volume data stored in the cloud with AWS KMS keys. Amazon EC2 Auto Scaling supports AWS managed and customer managed keys to encrypt your data. Note that the `KmsKeyId` option to specify a customer managed key is not available when you use a launch configuration. To specify your customer managed key, use a launch template instead. For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

You can also configure a customer managed key in your EBS-backed AMI before setting up the launch template or launch configuration, or use encryption by default to enforce the encryption of the new EBS volumes and snapshot copies that you create.

For information about how to set up the key policy that you need to launch Auto Scaling instances when you use a customer managed key for encryption, see [Required AWS KMS key policy for use with encrypted volumes \(p. 353\)](#). For information about how to create, store, and manage your AWS KMS encryption keys, see [What is AWS Key Management Service?](#)

Related topics

- [Data protection in Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Use encryption with EBS-backed AMIs](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Encryption by default](#) in the *Amazon EC2 User Guide for Linux Instances*

Identity and Access Management for Amazon EC2 Auto Scaling

AWS Identity and Access Management (IAM) is an 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 Amazon EC2 Auto Scaling resources. IAM is an AWS service that you can use with no additional charge.

To use Amazon EC2 Auto Scaling, you need an AWS account and your specific user credentials for signing into your account. For more information, see the [IAM User Guide](#).

After your account is set up, you will need to obtain your AWS access keys if you want to access Amazon EC2 Auto Scaling through its API, whether by the Query (HTTPS) interface directly or indirectly through an [SDK](#), the [AWS Command Line Interface](#), or the [AWS Tools for Windows PowerShell](#). AWS access keys consist of an access key ID and a secret access key. For more information about getting your AWS access keys, see [AWS security credentials](#) in the [AWS General Reference](#).

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon EC2 Auto Scaling resources. For example, you must have permissions to create Auto Scaling groups, create launch configurations, and so on.

The following sections provide details on how an IAM administrator can use IAM to help secure your Amazon EC2 Auto Scaling resources, by controlling who can perform Amazon EC2 Auto Scaling actions.

We recommend that you read the Amazon EC2 topics first. See [Identity and access management for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*. After reading the topics in this section, you should have a good idea what access control permissions Amazon EC2 offers and how they can fit in with your Amazon EC2 Auto Scaling resource permissions.

Topics

- [How Amazon EC2 Auto Scaling works with IAM \(p. 324\)](#)
- [AWS managed policies for Amazon EC2 Auto Scaling \(p. 330\)](#)
- [Service-linked roles for Amazon EC2 Auto Scaling \(p. 332\)](#)
- [Amazon EC2 Auto Scaling identity-based policy examples \(p. 337\)](#)
- [Cross-service confused deputy prevention \(p. 346\)](#)
- [Launch template support \(p. 347\)](#)
- [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#)
- [Required AWS KMS key policy for use with encrypted volumes \(p. 353\)](#)

How Amazon EC2 Auto Scaling works with IAM

Before you use IAM to manage access to Amazon EC2 Auto Scaling, learn what IAM features are available to use with Amazon EC2 Auto Scaling.

IAM features you can use with Amazon EC2 Auto Scaling

IAM feature	Amazon EC2 Auto Scaling support
Identity-based policies (p. 325)	Yes
Resource-based policies (p. 325)	No
Policy actions (p. 325)	Yes
Policy resources (p. 326)	Yes
Policy condition keys (service-specific) (p. 327)	Yes
ACLs (p. 329)	No
ABAC (tags in policies) (p. 329)	Partial
Temporary credentials (p. 329)	Yes
Service roles (p. 330)	Yes
Service-linked roles (p. 330)	Yes

To get a high-level view of how Amazon EC2 Auto Scaling and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for Amazon EC2 Auto Scaling

Supports identity-based policies	Yes
----------------------------------	-----

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for Amazon EC2 Auto Scaling

To view examples of Amazon EC2 Auto Scaling identity-based policies, see [Amazon EC2 Auto Scaling identity-based policy examples](#) (p. 337).

Resource-based policies within Amazon EC2 Auto Scaling

Supports resource-based policies	No
----------------------------------	----

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

Policy actions for Amazon EC2 Auto Scaling

Supports policy actions	Yes
-------------------------	-----

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

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

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

To see a list of Amazon EC2 Auto Scaling actions, see [Actions defined by Amazon EC2 Auto Scaling](#) in the [Service Authorization Reference](#).

Policy actions in Amazon EC2 Auto Scaling use the following prefix before the action:

```
autoscaling
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [  
    "autoscaling:action1",  
    "autoscaling:action2"  
]
```

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

```
"Action": "autoscaling:Describe*"
```

To view examples of Amazon EC2 Auto Scaling identity-based policies, see [Amazon EC2 Auto Scaling identity-based policy examples \(p. 337\)](#).

Policy resources for Amazon EC2 Auto Scaling

Supports policy resources	Yes
---------------------------	-----

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

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

You can use ARNs to identify the Auto Scaling groups and launch configurations that the IAM policy applies to.

An Auto Scaling group has the following ARN.

```
"Resource": "arn:aws:autoscaling:region:account-id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

A launch configuration has the following ARN.

```
"Resource": "arn:aws:autoscaling:region:account-id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

To specify an Auto Scaling group with the `CreateAutoScalingGroup` action, you must replace the `UUID` with a wildcard (*) as shown in the following example.

```
"Resource": "arn:aws:autoscaling:region:account-  
id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

To specify a launch configuration with the `CreateLaunchConfiguration` action, you must replace the UUID with a wildcard (*) as shown in the following example.

```
"Resource": "arn:aws:autoscaling:region:account-  
id:launchConfiguration:*:launchConfigurationName/lc-name"
```

For more information about Amazon EC2 Auto Scaling resource types and their ARNs, see [Resources defined by Amazon EC2 Auto Scaling](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions defined by Amazon EC2 Auto Scaling](#).

Not all Amazon EC2 Auto Scaling actions support resource-level permissions. For actions that don't support resource-level permissions, you must use a wildcard (*) as the resource.

The following Amazon EC2 Auto Scaling actions do not support resource-level permissions.

- `DescribeAccountLimits`
- `DescribeAdjustmentTypes`
- `DescribeAutoScalingGroups`
- `DescribeAutoScalingInstances`
- `DescribeAutoScalingNotificationTypes`
- `DescribeInstanceRefreshes`
- `DescribeLaunchConfigurations`
- `DescribeLifecycleHooks`
- `DescribeLifecycleHookTypes`
- `DescribeLoadBalancers`
- `DescribeLoadBalancerTargetGroups`
- `DescribeMetricCollectionTypes`
- `DescribeNotificationConfigurations`
- `DescribePolicies`
- `DescribeScalingActivities`
- `DescribeScalingProcessTypes`
- `DescribeScheduledActions`
- `DescribeTags`
- `DescribeTerminationPolicyTypes`
- `DescribeWarmPool`

Policy condition keys for Amazon EC2 Auto Scaling

Supports service-specific policy condition keys	Yes
---	-----

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

The `Condition` element (or `Condition block`) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

Amazon EC2 Auto Scaling defines its own set of condition keys and also supports using some global condition keys.

Amazon EC2 Auto Scaling supports the following condition keys that you can use in permission policies to determine who can access Amazon EC2 Auto Scaling:

- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/key-name: tag-value`
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`

The following condition keys are specific to create launch configuration requests:

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`
- `autoscaling:MetadataHttpTokens`
- `autoscaling:SpotPrice`

Amazon EC2 Auto Scaling also supports the following global condition keys that you can use to define permissions based on the tags in the request or present on the Auto Scaling group. For more information, see [Tag Auto Scaling groups and instances \(p. 91\)](#).

- `aws:RequestTag/key-name: tag-value`
- `aws:ResourceTag/key-name: tag-value`
- `aws:TagKeys: [tag-key, ...]`

To learn which Amazon EC2 Auto Scaling API actions you can use a condition key with, see [Actions defined by Amazon EC2 Auto Scaling](#) in the *Service Authorization Reference*. For more information about using Amazon EC2 Auto Scaling condition keys, see [Condition keys for Amazon EC2 Auto Scaling](#).

For examples of IAM policies you can use to control access, see the following topics:

- For examples that use condition keys to control access to actions on Auto Scaling groups and launch configurations, see [Tag Auto Scaling groups and instances \(p. 91\)](#) and [Amazon EC2 Auto Scaling identity-based policy examples \(p. 337\)](#).
- For additional examples, including an example that denies access to Auto Scaling groups if a launch configuration is specified in the request, see [Launch template support \(p. 347\)](#).

ACLs in Amazon EC2 Auto Scaling

Supports ACLs	No
---------------	----

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Amazon EC2 Auto Scaling

Supports ABAC (tags in policies)	Partial
----------------------------------	---------

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see [What is ABAC?](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

ABAC is possible for resources that support tags, but not everything supports tags. Launch configurations and scaling policies don't support tags, but Auto Scaling groups support tags.

For more information about tagging Auto Scaling groups, see [Tag Auto Scaling groups and instances \(p. 91\)](#).

To view an example of an identity-based policy for limiting access to an Auto Scaling group based on the tags on that group, see [Tags for security \(p. 95\)](#).

Using temporary credentials with Amazon EC2 Auto Scaling

Supports temporary credentials	Yes
--------------------------------	-----

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switching to a role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

Service roles for Amazon EC2 Auto Scaling

Supports service roles	Yes
------------------------	-----

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

When you create a lifecycle hook that notifies an Amazon SNS topic or Amazon SQS queue, you must specify a role to allow Amazon EC2 Auto Scaling to access Amazon SNS or Amazon SQS on your behalf. Use the IAM console to set up the service role for your lifecycle hook. The console helps you create a role with a sufficient set of permissions using a managed policy. For more information, see [Receive notifications using Amazon SNS \(p. 202\)](#) and [Receive notifications using Amazon SQS \(p. 202\)](#).

When you create an Auto Scaling group, you can optionally pass in a service role to allow Amazon EC2 instances to access other AWS services on your behalf. The service role for Amazon EC2 instances (also called the Amazon EC2 instance profile for a launch template or launch configuration) is a special type of service role that is assigned to every EC2 instance in an Auto Scaling group when the instance launches. You can use the IAM console and AWS CLI to create or edit this service role. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).

Warning

Changing the permissions for a service role might break Amazon EC2 Auto Scaling functionality. Edit service roles only when Amazon EC2 Auto Scaling provides guidance to do so.

Service-linked roles for Amazon EC2 Auto Scaling

Supports service-linked roles	Yes
-------------------------------	-----

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action 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.

For details about creating or managing Amazon EC2 Auto Scaling service-linked roles, see [Service-linked roles for Amazon EC2 Auto Scaling \(p. 332\)](#).

AWS managed policies for Amazon EC2 Auto Scaling

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies.

These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

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

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

Amazon EC2 Auto Scaling managed policies

You can attach the following managed policies to your AWS Identity and Access Management (IAM) entities. Each policy provides access to all or some of the API actions for Amazon EC2 Auto Scaling.

- [AutoScalingFullAccess](#) — Grants full access to Amazon EC2 Auto Scaling for users who need full Amazon EC2 Auto Scaling access from the AWS CLI or SDKs, but not AWS Management Console access.
- [AutoScalingReadOnlyAccess](#) — Grants read-only access to Amazon EC2 Auto Scaling for users who are making calls only to the AWS CLI or SDKs.
- [AutoScalingConsoleFullAccess](#) — Grants full access to Amazon EC2 Auto Scaling using the AWS Management Console. This policy works when you are using launch configurations, but not when you are using launch templates.
- [AutoScalingConsoleReadOnlyAccess](#) — Grants read-only access to Amazon EC2 Auto Scaling using the AWS Management Console. This policy works when you are using launch configurations, but not when you are using launch templates.

When you are using launch templates from the console, you need to grant additional permissions specific to launch templates, which are discussed in [Launch template support \(p. 347\)](#). The Amazon EC2 Auto Scaling console needs permissions for `ec2` actions so it can display information about launch templates and launch instances using launch templates.

AutoScalingServiceRolePolicy AWS managed policy

You can't attach [AutoScalingServiceRolePolicy](#) to your IAM entities. This policy is attached to a service-linked role that allows Amazon EC2 Auto Scaling to launch and terminate instances. For more information, see [Service-linked roles for Amazon EC2 Auto Scaling \(p. 332\)](#).

Amazon EC2 Auto Scaling updates to AWS managed policies

View details about updates to AWS managed policies for Amazon EC2 Auto Scaling since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon EC2 Auto Scaling Document history page.

Change	Description	Date
Amazon EC2 Auto Scaling adds permissions to its service-linked role	The <code>AutoScalingServiceRolePolicy</code> policy now grants permission to	March 28, 2022

Change	Description	Date
	call the AWS Systems Manager GetParameters API action to support an upcoming feature to read parameters from Parameter Store. For more information, see Service-linked roles for Amazon EC2 Auto Scaling (p. 332) .	
Amazon EC2 Auto Scaling adds permissions to its service-linked role	To support predictive scaling, the <code>AutoScalingServiceRolePolicy</code> policy now includes permission to call the CloudWatch GetMetricData API action. For more information, see Service-linked roles for Amazon EC2 Auto Scaling .	May 19, 2021
Amazon EC2 Auto Scaling started tracking changes	Amazon EC2 Auto Scaling started tracking changes for its AWS managed policies.	May 19, 2021

Service-linked roles for Amazon EC2 Auto Scaling

Amazon EC2 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 IAM role that is linked directly to an AWS service.

Service-linked roles provide a secure way to delegate permissions to other 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*. Service-linked roles also enable all API calls to be visible through AWS CloudTrail. This helps with monitoring and auditing requirements because you can track all actions that Amazon EC2 Auto Scaling performs on your behalf. For more information, see [Log Amazon EC2 Auto Scaling API calls with AWS CloudTrail \(p. 269\)](#).

The following sections describe how to create and manage Amazon EC2 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. For more information, see [Using service-linked roles](#) in the *IAM User Guide*.

Contents

- [Overview \(p. 332\)](#)
- [Permissions granted by the service-linked role \(p. 333\)](#)
- [Create a service-linked role \(automatic\) \(p. 335\)](#)
- [Create a service-linked role \(manual\) \(p. 335\)](#)
- [Edit the service-linked role \(p. 336\)](#)
- [Delete the service-linked role \(p. 336\)](#)
- [Supported Regions for Amazon EC2 Auto Scaling service-linked roles \(p. 337\)](#)

Overview

There are two types of Amazon EC2 Auto Scaling service-linked roles:

- The default service-linked role for your account, named `AWSServiceRoleForAutoScaling`. This role is automatically assigned to your Auto Scaling groups unless you specify a different service-linked role.
 - A service-linked role with a custom suffix that you specify when you create the role, for example, `AWSServiceRoleForAutoScaling_mysuffix`.

The permissions of a custom suffix service-linked role are identical to those of the default service-linked role. In both cases, you cannot edit the roles, and you also cannot delete them if they are still in use by an Auto Scaling group. The only difference is the role name suffix.

You can specify either role when you edit your AWS Key Management Service key policies to allow instances that are launched by Amazon EC2 Auto Scaling to be encrypted with your customer managed key. However, if you plan to give granular access to a specific customer managed key, you should use a custom suffix service-linked role. Using a custom suffix service-linked role provides you with:

- More control over the customer managed key
 - The ability to track which Auto Scaling group made an API call in your CloudTrail logs

If you create customer managed keys that not all users should have access to, follow these steps to allow the use of a custom suffix service-linked role:

1. Create a service-linked role with a custom suffix. For more information, see [Create a service-linked role \(manual\) \(p. 335\)](#).
 2. Give the service-linked role access to a customer managed key. For more information about the key policy that allows the key to be used by a service-linked role, see [Required AWS KMS key policy for use with encrypted volumes \(p. 353\)](#).
 3. Give IAM users or roles access to the service-linked role that you created. For more information about creating the IAM policy, see [Control which service-linked role can be passed \(using PassRole\) \(p. 344\)](#). If users try to specify a service-linked role without permission to pass that role to the service, they receive an error.

Permissions granted by the service-linked role

Amazon EC2 Auto Scaling uses the `AWSServiceRoleForAutoScaling` service-linked role or your custom suffix service-linked role to call AWS APIs on your behalf:

The role permissions policy allows Amazon EC2 Auto Scaling to complete the following actions on resources.

```

        "ec2:TerminateInstances"
    ],
    "Resource": "*"
},
{
    "Sid": "EC2InstanceProfileManagement",
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "iam:PassedToService": "ec2.amazonaws.com*"
        }
    }
},
{
    "Sid": "EC2SpotManagement",
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "iam:AWSServiceName": "spot.amazonaws.com"
        }
    }
},
{
    "Sid": "ELBManagement",
    "Effect": "Allow",
    "Action": [
        "elasticloadbalancing:Register*",
        "elasticloadbalancing:Deregister*",
        "elasticloadbalancing:Describe*"
    ],
    "Resource": "*"
},
{
    "Sid": "CWManagement",
    "Effect": "Allow",
    "Action": [
        "cloudwatch:DeleteAlarms",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricData",
        "cloudwatch:PutMetricAlarm"
    ],
    "Resource": "*"
},
{
    "Sid": "SNSManagement",
    "Effect": "Allow",
    "Action": [
        "sns:Publish"
    ],
    "Resource": "*"
},
{
    "Sid": "EventBridgeRuleManagement",
    "Effect": "Allow",
    "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",

```

```
        "events:DeleteRule",
        "events:DescribeRule"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "events:ManagedBy": "autoscaling.amazonaws.com"
        }
    }
},
{
    "Sid": "SystemsManagerParameterManagement",
    "Effect": "Allow",
    "Action": [
        "ssm:GetParameters"
    ],
    "Resource": "*"
}
]
}
```

The role trusts the `autoscaling.amazonaws.com` service to assume it.

Create a service-linked role (automatic)

Amazon EC2 Auto Scaling creates the `AWSServiceRoleForAutoScaling` service-linked role for you the first time that you create an Auto Scaling group, unless you manually create a custom suffix service-linked role and specify it when creating the group.

Important

You must have 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* and [Required permissions to create a service-linked role \(p. 344\)](#) in this guide.

Amazon EC2 Auto Scaling began supporting service-linked roles in March 2018. If you created an Auto Scaling group before then, Amazon EC2 Auto Scaling created the `AWSServiceRoleForAutoScaling` role in your account. For more information, see [A new role appeared in my AWS account](#) in the *IAM User Guide*.

Create a service-linked role (manual)

To create a service-linked role (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**, **Create role**.
3. For **Select trusted entity**, choose **AWS service**.
4. For **Choose the service that will use this role**, choose **EC2 Auto Scaling** and the **EC2 Auto Scaling** use case.
5. Choose **Next: Permissions**, **Next: Tags**, and then **Next: Review**. Note: You cannot attach tags to service-linked roles during creation.
6. On the **Review** page, leave **Role name** blank to create a service-linked role with the name `AWSServiceRoleForAutoScaling`, or enter a suffix to create a service-linked role with the name `AWSServiceRoleForAutoScaling_suffix`.
7. (Optional) For **Role description**, edit the description for the service-linked role.
8. Choose **Create role**.

To create a service-linked role (AWS CLI)

Use the following [create-service-linked-role](#) CLI command to create a service-linked role for Amazon EC2 Auto Scaling with the name `AWSServiceRoleForAutoScaling_suffix`.

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --custom-suffix suffix
```

The output of this command includes the ARN of the service-linked role, which you can use to give the service-linked role access to your customer managed key.

```
{  
  "Role": {  
    "RoleId": "ABCDEF0123456789ABCDEF",  
    "CreateDate": "2018-08-30T21:59:18Z",  
    "RoleName": "AWSServiceRoleForAutoScaling_suffix",  
    "Arn": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",  
    "Path": "/aws-service-role/autoscaling.amazonaws.com/",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Action": [  
            "sts:AssumeRole"  
          ],  
          "Principal": {  
            "Service": [  
              "autoscaling.amazonaws.com"  
            ]  
          },  
          "Effect": "Allow"  
        }  
      ]  
    }  
  }  
}
```

For more information, see [Creating a service-linked role](#) in the *IAM User Guide*.

Edit the service-linked role

You cannot edit the service-linked roles that are created for Amazon EC2 Auto Scaling. After you create a service-linked role, you cannot change the name of the role or its permissions. However, you can edit the description of the role. For more information, see [Editing a service-linked role](#) in the *IAM User Guide*.

Note

In AWS, modifying service-linked roles is discouraged because it can lead to security issues such as the [cross-service confused deputy](#).

Delete the service-linked role

If you are not using an Auto Scaling group, we recommend that you delete its service-linked role. Deleting the role prevents you from having an entity that is not used or actively monitored and maintained.

You can delete a service-linked role only after first deleting the related dependent resources. This protects you from inadvertently revoking Amazon EC2 Auto Scaling permissions to your resources. If a service-linked role is used with multiple Auto Scaling groups, you must delete all Auto Scaling groups that use the service-linked role before you can delete it. For more information, see [Delete your Auto Scaling infrastructure \(p. 117\)](#).

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

If you delete the `AWSServiceRoleForAutoScaling` service-linked role, Amazon EC2 Auto Scaling creates the role again when you create an Auto Scaling group and do not specify a different service-linked role.

Supported Regions for Amazon EC2 Auto Scaling service-linked roles

Amazon EC2 Auto Scaling supports using service-linked roles in all of the AWS Regions where the service is available.

Amazon EC2 Auto Scaling identity-based policy examples

By default, a brand new IAM user has no permissions to do anything. An IAM administrator must create and assign IAM policies that give end users permission to perform Amazon EC2 Auto Scaling API actions.

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 shows an example of a permissions policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "autoscaling:CreateAutoScalingGroup",  
                "autoscaling:UpdateAutoScalingGroup",  
                "autoscaling:DeleteAutoScalingGroup"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": { "autoscaling:ResourceTag/environment": "test" }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "autoscaling:*LaunchConfiguration*",  
                "autoscaling:Describe"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

This sample policy gives users permissions to create, modify, and delete Auto Scaling groups, but only if the group uses the tag `environment=test`. Because launch configurations do not support tags, and `Describe` actions do not support resource-level permissions, you must specify them in a separate statement without conditions. To learn more about the elements within an IAM policy statement, see [Identity-based policies for Amazon EC2 Auto Scaling \(p. 325\)](#).

Contents

- [Policy best practices \(p. 338\)](#)
- [Customer managed policy examples \(p. 339\)](#)

- Control which tag keys and tag values can be used (p. 339)
- Control access to Auto Scaling resources based on tags (p. 340)
 - Control access to creating and managing Auto Scaling groups and scaling policies (p. 340)
 - Control which scaling policies can be deleted (p. 341)
- Control the minimum and maximum capacity of Auto Scaling groups (p. 341)
- Control which IAM roles can be passed (using `PassRole`) (p. 342)
- Allow users to change the capacity of Auto Scaling groups (p. 342)
- Allow users to create and use launch configurations (p. 343)
- Allow users to create and use launch templates (p. 344)
- Required permissions to create a service-linked role (p. 344)
 - Control which service-linked role can be passed (using `PassRole`) (p. 344)
- Required API permissions for Amazon EC2 Auto Scaling (p. 345)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon EC2 Auto Scaling resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [IAM Access Analyzer policy validation](#) in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or root users in your account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Configuring MFA-protected API access](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Note

Some Amazon EC2 Auto Scaling API actions allow you to include specific Auto Scaling groups in your policy that can be created or modified by the action. You can restrict the target resources

for these actions by specifying individual Auto Scaling group ARNs. As a best practice, however, we recommend that you use tag-based policies that allow (or deny) actions on Auto Scaling groups with a specific tag.

Customer managed policy examples

You can create your own custom IAM policies to allow or deny permissions for IAM users or groups to perform Amazon EC2 Auto Scaling actions. You can attach these custom policies to the IAM users or groups that require the specified permissions. The following examples show permissions for several common use cases.

If you are new to creating policies, we recommend that you first create an IAM user in your account and attach policies to the user. You can use the console to verify the effects of each policy as you attach the policy to the user.

When creating and updating Auto Scaling groups, some actions require that certain other actions be carried out. You can specify these other actions in the Action element of an IAM policy statement. For example, there are additional API actions for Elastic Load Balancing, CloudWatch, and Amazon SNS that might be required depending on the access that you want to provide for a user.

Topics

- [Control which tag keys and tag values can be used \(p. 339\)](#)
- [Control access to Auto Scaling resources based on tags \(p. 340\)](#)
- [Control the minimum and maximum capacity of Auto Scaling groups \(p. 341\)](#)
- [Control which IAM roles can be passed \(using PassRole\) \(p. 342\)](#)
- [Allow users to change the capacity of Auto Scaling groups \(p. 342\)](#)
- [Allow users to create and use launch configurations \(p. 343\)](#)
- [Allow users to create and use launch templates \(p. 344\)](#)

Control which tag keys and tag values can be used

You can use conditions in your IAM policies to control the tag keys and tag values that can be applied to Auto Scaling groups.

To give users permissions to create or tag an Auto Scaling group only if they specify certain tags, use the `aws:RequestTag` condition key. To allow only specific tag keys, use the `aws:TagKeys` condition key with the `ForAllValues` modifier.

The following policy requires users to specify a tag with the key `environment` in the request. The `"?*` value enforces that there is some value for the tag key. To use a wildcard, you must use the `StringLike` condition operator.

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": [  
            "autoscaling:CreateAutoScalingGroup",  
            "autoscaling:CreateOrUpdateTags"  
        ],  
        "Resource": "*",  
        "Condition": {  
            "StringLike": { "aws:RequestTag/environment": "?*" }  
        }  
    }]  
}
```

The following policy specifies that users can only tag Auto Scaling groups with the tags `purpose=webserver` and `cost-center=cc123`, and allows only the `purpose` and `cost-center` tags (no other tags can be specified).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/purpose": "webserver",
          "aws:RequestTag/cost-center": "cc123"
        },
        "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
      }
    }
  ]
}
```

The following policy requires users to specify at least one tag in the request, and allows only the `cost-center` and `owner` keys.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
      }
    }
  ]
}
```

Note

For conditions, the condition key is not case-sensitive and the condition value is case-sensitive. Therefore, to enforce the case-sensitivity of a tag key, use the `aws:TagKeys` condition key, where the tag key is specified as a value in the condition.

Control access to Auto Scaling resources based on tags

You can also provide tag information in your IAM policies to control access based on the tags that are attached to the Auto Scaling group by using the `autoscaling:ResourceTag` condition key.

Control access to creating and managing Auto Scaling groups and scaling policies

The following policy gives users permissions to use all Amazon EC2 Auto Scaling actions that include the string `Scaling` in their names, as long as the Auto Scaling group has the tag `purpose=webserver`. Because the `Describe` actions do not support resource-level permissions, you must specify them in a separate statement without conditions.

```
{
  "Version": "2012-10-17",
  "Statement": [

```

```
{
  "Effect": "Allow",
  "Action": ["autoscaling:*Scaling*"],
  "Resource": "*",
  "Condition": {
    "StringEquals": { "autoscaling:ResourceTag/purpose": "webserver" }
  }
},
{
  "Effect": "Allow",
  "Action": "autoscaling:Describe*Scaling*",
  "Resource": "*"
}
]
```

Control which scaling policies can be deleted

The following policy allows users to use the `autoscaling:DeletePolicy` action to delete a scaling policy. However, it also denies the action if the Auto Scaling group being acted upon has the tag `environment=production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*",
      "Condition": {
        "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
      }
    }
  ]
}
```

Control the minimum and maximum capacity of Auto Scaling groups

Amazon EC2 Auto Scaling allows you to restrict the size of the Auto Scaling groups that can be created. The following policy gives users permissions to create and update all Auto Scaling groups with the tag `allowed=true`, as long as they don't specify a minimum size less than 1 or a maximum size greater than 10.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": { "autoscaling:ResourceTag/allowed": "true" },
        "NumericGreaterThanOrEqualIfExists": { "autoscaling:MinSize": 1 },
        "NumericLessThanOrEqualIfExists": { "autoscaling:MaxSize": 10 }
      }
    }
  ]
}
```

}

Control which IAM roles can be passed (using PassRole)

If you want a user to be able to create Amazon EC2 Auto Scaling resources that specify an instance profile (a container for an IAM role), you must use a policy that includes a statement allowing the user to pass the role, like the following example. By specifying the ARN, the policy grants the user the permission to pass only roles whose name begins with `qateam-`. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam::123456789012:role/qateam-*",  
            "Condition": {  
                "StringEquals": {  
                    "iam:PassedToService": [  
                        "ec2.amazonaws.com",  
                        "ec2.amazonaws.com.cn"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Allow users to change the capacity of Auto Scaling groups

The following policy gives users permissions to use the `SetDesiredCapacity` and `TerminateInstanceInAutoScalingGroup` API actions. The `Resource` element uses a wildcard (*) to indicate that users can change the capacity of any Auto Scaling group.

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

If you are not using tags to control access to Auto Scaling groups, you can adjust the preceding statement to give users permissions to change the capacity of only Auto Scaling groups whose name begins with `devteam-`. For more information about specifying the ARN value, see [Resource-based policies within Amazon EC2 Auto Scaling \(p. 325\)](#).

```
        "Resource":  
        "arn:aws:autoscaling:region:123456789012:autoScalingGroup:*:autoScalingGroupName/devteam-*"
```

You can also specify multiple ARNs by enclosing them in a list. Including the UUID ensures that access is granted to the specific Auto Scaling group. The UUID for a new group is different than the UUID for a deleted group with the same name.

```

"Resource": [
    "arn:aws:autoscaling:region:123456789012:autoScalingGroup:7fe02b8e-7442-4c9e-8c8e-85fa99e9b5d9:autoScal
    devteam-1",
    "arn:aws:autoscaling:region:123456789012:autoScalingGroup:9d8e8ea4-22e1-44c7-
    a14d-520f8518c2b9:autoScalingGroupName/devteam-2",
    "arn:aws:autoscaling:region:123456789012:autoScalingGroup:60d6b363-
    ae8b-467c-947f-f1d308935521:autoScalingGroupName/devteam-3"
]

```

Allow users to create and use launch configurations

The following policy gives users permissions to create a launch configuration if the instance type is `t2.micro`, but only if the name of the launch configuration starts with `qateam-`. For more information about specifying the ARN value, see [Resource-based policies within Amazon EC2 Auto Scaling \(p. 325\)](#). They can specify a launch configuration for an Auto Scaling group only if its name starts with `qateam-`.

The last part of the statement gives users permissions to describe launch configurations and to access certain Amazon EC2 resources in their account. This gives users minimum permissions to create and manage launch configurations from the Amazon EC2 Auto Scaling console.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateLaunchConfiguration",
            "Resource": "arn:aws:autoscaling:region:123456789012:launchConfiguration:*qateam-*",
            "Condition": {
                "StringEquals": { "autoscaling:InstanceType": "t2.micro" }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling>CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": { "autoscaling:LaunchConfigurationName": "qateam-*" }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeLaunchConfigurations",
                "ec2:DescribeImages",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        }
    ]
}

```

```
}
```

You can add API actions to this policy to provide more options for users, for example:

- `iam:ListInstanceProfiles`: To list instance profiles.
- `ec2:CreateSecurityGroup`: To create a new security group.
- `ec2:AuthorizeSecurityGroupIngress`: To add inbound rules.
- `ec2:CreateKeyPair`: To create a new key pair.

Allow users to create and use launch templates

For example policies, see [Launch template support \(p. 347\)](#).

Required permissions to create a service-linked role

Amazon EC2 Auto Scaling requires permissions to create a service-linked role the first time that any user in your AWS account calls Amazon EC2 Auto Scaling API actions. If the service-linked role does not exist already, Amazon EC2 Auto Scaling creates it in your account. The service-linked role gives permissions to Amazon EC2 Auto Scaling so that it can call other AWS 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 Amazon EC2 Auto Scaling service-linked role for Amazon EC2 Auto Scaling.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

Control which service-linked role can be passed (using `PassRole`)

If your users require the ability to pass custom suffix service-linked roles to an Auto Scaling group, you must attach a policy to the users or roles, based on the access that they need. We recommend that you restrict this policy to only the service-linked roles that your users must access. For more information about custom suffix service-linked roles, see [Service-linked roles for Amazon EC2 Auto Scaling \(p. 332\)](#).

The following example is helpful for facilitating the security of your AWS KMS customer managed keys if you give different service-linked roles access to different keys. Depending on your needs, you might have a key for the development team, another for the QA team, and another for the finance team. First, create a service-linked role that has access to the required key, for example, a service-linked role named `AWSServiceRoleForAutoScaling_devteamkeyaccess`. Then, to grant permissions to pass that service-linked role to an Auto Scaling group, attach the policy to your IAM users as shown.

The policy in this example gives users permissions to pass the `AWSServiceRoleForAutoScaling_devteamkeyaccess` role to create any Auto Scaling group whose name begins with `devteam-`. If they try to specify a different service-linked role, they receive an error. If they choose not to specify a service-linked role, the default `AWSServiceRoleForAutoScaling` role is used instead.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/aws-service-role/autScaling.amazonaws.com/AWSServiceRoleForAutoScaling_devteamkeyaccess",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "autScaling.amazonaws.com"
          ]
        },
        "StringLike": {
          "iam:AssociatedResourceARN": [
            "arn:aws:autScaling:region:123456789012:autoScalingGroup:*:autoScalingGroupName/devteam-*"
          ]
        }
      }
    }
  ]
}
```

Required API permissions for Amazon EC2 Auto Scaling

When calling the following actions from the Amazon EC2 Auto Scaling API, users must have permissions from Amazon EC2 and IAM to perform certain actions. You specify the following actions in the `Action` element of an IAM policy statement.

Create an Auto Scaling group

- `autScaling:CreateAutoScalingGroup`
- `iam:CreateServiceLinkedRole` (Needed if you are using the default service-linked role and that role does not yet exist)
- `iam:PassRole` (Needed if you are using a nondefault service-linked role, you are specifying an IAM role for the `RoleARN` parameter for a lifecycle hook, or you are using a launch template that specifies the IAM role that can be used by the instances in the Auto Scaling group)
- `ec2:RunInstances` (Needed if you are using a launch template)
- `ec2:CreateTags` (Needed if you are using a launch template that specifies the tags to apply to your instances and EBS volumes)

Create a launch configuration

- `autScaling:CreateLaunchConfiguration`
- `ec2:DescribeImages`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeKeyPairs`

- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeVpcClassicLink`
- `iam:PassRole` (Needed if you are specifying an IAM role for the `IamInstanceProfile` parameter)

Create a lifecycle hook

- `autoscaling:PutLifecycleHook`
- `iam:PassRole` (Only needed if you are specifying an IAM role for the `RoleARN` parameter)

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action.

In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access.

To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account. We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in trust policies for Amazon EC2 Auto Scaling service roles. These keys limit the permissions that Amazon EC2 Auto Scaling gives another service to the resource.

To use the `aws:SourceArn` or `aws:SourceAccount` global condition keys, set the value to the Amazon Resource Name (ARN) or account of the resource that Amazon EC2 Auto Scaling stores. Whenever possible, use `aws:SourceArn`, which is more specific. Set the value to the ARN or an ARN pattern with wildcards (*) for the unknown portions of the ARN. If you don't know the ARN of the resource, use `aws:SourceAccount` instead.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Amazon EC2 Auto Scaling to prevent the confused deputy problem.

Example: Using `aws:SourceArn` and `aws:SourceAccount` condition keys

A role that a service assumes to perform actions on your behalf is called a [service role \(p. 330\)](#). In cases where you want to create lifecycle hooks that send notifications to anywhere other than Amazon EventBridge, you must create a service role to allow Amazon EC2 Auto Scaling to send notifications to an Amazon SNS topic or Amazon SQS queue on your behalf. If you want only one Auto Scaling group to be associated with the cross-service access, you can specify the trust policy of the service role as follows.

This example trust policy uses condition statements to limit the `AssumeRole` capability on the service role to only the actions that affect the specified Auto Scaling group in the specified account. The `aws:SourceArn` and `aws:SourceAccount` conditions are evaluated independently. Any request to use the service role must satisfy both conditions.

Before using this policy, replace the Region, account ID, UUID, and group name with valid values from your account.

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
```

```

    "Effect": "Allow",
    "Principal": {
        "Service": "autoscaling.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn":
                "arn:aws:autoscaling:region:account_id:autoScalingGroup:uuid:autoScalingGroupName/my-asg"
        },
        "StringEquals": {
            "aws:SourceAccount": "account_id"
        }
    }
}
}

```

Additional information

For more information, see [AWS global condition context keys](#), [The confused deputy problem](#), and [Modifying a role trust policy \(console\)](#) in the *IAM User Guide*.

Launch template support

Amazon EC2 Auto Scaling supports using Amazon EC2 launch templates with your Auto Scaling groups. We recommend that you allow users to create Auto Scaling groups from launch templates, because doing so allows them to use the latest features of Amazon EC2 Auto Scaling and Amazon EC2. For example, users must specify a launch template to use a [mixed instances policy](#).

You can use the `AmazonEC2FullAccess` policy to give users complete access to work with Amazon EC2 Auto Scaling resources, launch templates, and other EC2 resources in their account. Or, you can create your own custom IAM policies to give users fine-grained permissions to work with launch templates, as described in this topic.

A sample policy that you can tailor for your own use

The following shows an example of a permissions policy that you can tailor for your own use. The policy allows IAM users to create, modify, and delete all Auto Scaling groups, but only if the group uses the tag `environment=test`. It then gives permission for all `Describe` actions. Because `Describe` actions do not support resource-level permissions, you must specify them in a separate statement without conditions.

Users with this policy have permission to create or update an Auto Scaling group using a launch template because they're also given permission to use the `ec2:RunInstances` action.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": { "autoscaling:ResourceTag/environment": "test" }
            }
        },
        {
            "Effect": "Allow",
            "Action": [

```

```
        "autoscaling:Describe*",
        "ec2:RunInstances"
    ],
    "Resource": "*"
}
]
```

ec2:RunInstances are checked when an Auto Scaling group is created or updated using a launch template. If you want to restrict access to the resources that are used to launch an instance or otherwise limit what IAM users can do, you must modify this policy to add your own statements that filter these permissions.

The following examples show policy statements that you could use to control the permissions that users have when using launch templates.

Contents

- [Require launch templates that have a specific tag \(p. 348\)](#)
- [Require a launch template and a version number \(p. 348\)](#)
- [Require the use of instance metadata service version 2 \(IMDSv2\) \(p. 349\)](#)
- [Restrict access to Amazon EC2 resources \(p. 349\)](#)
- [Permissions required to tag instances and volumes \(p. 350\)](#)
- [Additional permissions \(p. 350\)](#)
- [Learn more \(p. 351\)](#)

Require launch templates that have a specific tag

The following example restricts access to calling the ec2:RunInstances action with launch templates that are located in the specified Region and that have the tag environment=test.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:123456789012:launch-template/*",
            "Condition": {
                "StringEquals": { "ec2:ResourceTag/environment": "test" }
            }
        }
    ]
}
```

Require a launch template and a version number

The following example allows users to create and modify Auto Scaling groups if they specify the version number of the launch template, and then denies permission to create or modify any Auto Scaling groups using a launch configuration. If users with this policy omit the version number to specify either the Latest or Default launch template version, or specify a launch configuration instead, the action fails.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [

```

```

        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
    }
},
{
    "Effect": "Deny",
    "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
    ],
    "Resource": "*",
    "Condition": {
        "Null": { "autoscaling:LaunchConfigurationName": "false" }
    }
}
]
}

```

Require the use of instance metadata service version 2 (IMDSv2)

For extra security, you can set your users' permissions to require the use of a launch template that requires IMDSv2. For more information, see [Configuring the instance metadata service](#) in the *Amazon EC2 User Guide for Linux Instances*.

The following example specifies that users can't call the `ec2:RunInstances` action unless the instance is also opted in to require the use of IMDSv2 (indicated by `"ec2:MetadataHttpTokens": "required"`).

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:instance/*",
            "Condition": {
                "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
            }
        }
    ]
}

```

Tip

To force replacement Auto Scaling instances to launch that use a new launch template or a new version of a launch template with the instance metadata options configured, you can terminate existing instances in the group. Amazon EC2 Auto Scaling immediately starts launching new instances to replace the instances that you terminated. Alternatively, you can start an instance refresh to do a rolling update of your group. For more information, see [Replace Auto Scaling instances based on an instance refresh \(p. 99\)](#).

Restrict access to Amazon EC2 resources

The following example controls the configuration of the instances that a user can launch by restricting access to Amazon EC2 resources.

```
{
}
```

```

"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": "ec2:RunInstances",
        "Resource": [
            "arn:aws:ec2:region:123456789012:subnet/subnet-1a2b3c4d",
            "arn:aws:ec2:region:123456789012:security-group/sg-903004f88example",
            "arn:aws:ec2:region:123456789012:network-interface/*",
            "arn:aws:ec2:region:123456789012:volume/*",
            "arn:aws:ec2:region::image/ami-04d5cc9b88example"
        ]
    },
    {
        "Effect": "Allow",
        "Action": "ec2:RunInstances",
        "Resource": "arn:aws:ec2:region:123456789012:instance/*",
        "Condition": {
            "StringEquals": { "ec2:InstanceType": "t2.micro" }
        }
    }
]
}

```

In this example, there are two statements:

- The first statement requires that users launch instances into a specific subnet (subnet-1a2b3c4d), using a specific security group (sg-903004f88example), and using a specific AMI (ami-04d5cc9b88example). It also gives users access to additional resources that they need to launch instances: network interfaces and volumes.
- The second statement allows users to launch instances only of a specific instance type (t2.micro).

Permissions required to tag instances and volumes

The following example allows users to tag instances and volumes on creation. This policy is needed if there are tags specified in the launch template. For more information, see [Grant permission to tag resources during creation](#) in the *Amazon EC2 User Guide for Linux Instances*.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:region:123456789012:/*",
            "Condition": {
                "StringEquals": { "ec2:CreateAction": "RunInstances" }
            }
        }
    ]
}

```

Additional permissions

Depending on which scenarios you want to support, you can specify these additional actions in the Action element of an IAM policy statement.

For a user to have the ability to pass an IAM role to provisioned instances, that user must have permission for `iam:PassRole`. You can use an `iam:PassRole` policy to allow (or deny) users to pass a

role to Amazon EC2 if the launch template specifies an instance profile with an IAM role. For an example policy, see [Control which IAM roles can be passed \(using PassRole\) \(p. 342\)](#).

You must give your console users permissions for the `ec2:DescribeLaunchTemplates` and `ec2:DescribeLaunchTemplateVersions` actions. Without these permissions, launch template data cannot load in the Auto Scaling group wizard, and users cannot step through the wizard to launch instances using a launch template.

To control access to the `ec2:CreateLaunchTemplate` and `ec2:CreateLaunchTemplateVersion` actions, see [Control the use of launch templates](#) and [Example: Work with launch templates](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

For groups that are configured to use the `Latest` or `Default` launch template version, permissions for actions to be completed when launching instances are not checked by Amazon EC2 Auto Scaling when a new version of the launch template is created. This is an important consideration when setting up your permissions for who can create and manage launch template versions.

Learn more

Before you use IAM to manage access to launch templates, you should understand what IAM features are available to use with launch templates. For more information, see [Actions, resources, and condition keys for Amazon EC2](#) in the *Service Authorization Reference*.

IAM role for applications that run on Amazon EC2 instances

Applications that run on Amazon EC2 instances need credentials to access other AWS services. To provide these credentials in a secure way, use an IAM role. The role supplies temporary permissions that the application can use when it accesses other AWS resources. The role's permissions determine what the application is allowed to do.

For instances in an Auto Scaling group, you must create a launch template or launch configuration and choose an instance profile to associate with the instances. An instance profile is a container for an IAM role that allows Amazon EC2 to pass the IAM role to an instance when the instance is launched. First, create an IAM role that has all of the permissions required to access the AWS resources. Then, create the instance profile and assign the role to it.

Note

As a best practice, we strongly recommend that you create the role so that it has the minimum permissions to other AWS services that your application requires.

Contents

- [Prerequisites \(p. 351\)](#)
- [Create a launch template \(p. 352\)](#)
- [Create a launch configuration \(p. 352\)](#)
- [See also \(p. 353\)](#)

Prerequisites

Create the IAM role that your application running on Amazon EC2 can assume. Choose the appropriate permissions so that the application that is subsequently given the role can make the specific API calls that it needs.

If you use the IAM console instead of the AWS CLI or one of the AWS SDKs, the console creates an instance profile automatically and gives it the same name as the role to which it corresponds.

To create an IAM role (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane on the left, choose **Roles**.
3. Choose **Create role**.
4. For **Select trusted entity**, choose **AWS service**.
5. For your use case, choose **EC2** and then choose **Next**.
6. If possible, select the policy to use for the permissions policy or choose **Create policy** to open a new browser tab and create a new policy from scratch. For more information, see [Creating IAM policies](#) in the *IAM User Guide*. After you create the policy, close that tab and return to your original tab. Select the check box next to the permissions policies that you want the service to have.
7. (Optional) Set a permissions boundary. This is an advanced feature that is available for service roles. For more information, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
8. Choose **Next**.
9. On the **Name, review, and create** page, for **Role name**, enter a role name to help you identify the purpose of this role. This name must be unique within your AWS account. Because other AWS resources might reference the role, you can't edit the name of the role after it has been created.
10. Review the role, and then choose **Create role**.

Use IAM user permissions to control access to your new IAM role. The `iam:PassRole` permission is needed on the IAM user who creates or updates an Auto Scaling group using a launch template that specifies an instance profile, or who creates a launch configuration that specifies an instance profile. For an example policy that grants users specific permissions, see [Control which IAM roles can be passed \(using PassRole\)](#) (p. 342).

Create a launch template

When you create the launch template using the AWS Management Console, in the **Advanced details** section, select the role from **IAM instance profile**. For more information, see [Configure advanced settings for your launch template](#) (p. 27).

When you create the launch template using the `create-launch-template` command from the AWS CLI, specify the instance profile name of your IAM role as shown in the following example.

```
aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --  
version-description version1 \  
--launch-template-data  
'{"ImageId": "ami-04d5cc9b88example", "InstanceType": "t2.micro", "IamInstanceProfile":  
{"Name": "my-instance-profile"}'}
```

Create a launch configuration

When you create the launch configuration using the AWS Management Console, in the **Additional configuration** section, select the role from **IAM instance profile**. For more information, see [Create a launch configuration](#) (p. 40).

When you create the launch configuration using the `create-launch-configuration` command from the AWS CLI, specify the instance profile name of your IAM role as shown in the following example.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-with-  
instance-profile \  
'
```

```
--image-id ami-04d5cc9b88example --instance-type t2.micro \  
--iam-instance-profile my-instance-profile
```

See also

For more information to help you start learning about and using IAM roles for Amazon EC2, see:

- [IAM roles for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*
- [Using instance profiles](#) and [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*

Required AWS KMS key policy for use with encrypted volumes

Amazon EC2 Auto Scaling uses [service-linked roles \(p. 332\)](#) to delegate permissions to other AWS services. Amazon EC2 Auto Scaling service-linked roles are predefined and include permissions that Amazon EC2 Auto Scaling requires to call other AWS services on your behalf. The predefined permissions also include access to your AWS managed keys. However, they do not include access to your customer managed keys, allowing you to maintain full control over these keys.

This topic describes how to set up the key policy that you need to launch Auto Scaling instances when you specify a customer managed key for Amazon EBS encryption.

Note

Amazon EC2 Auto Scaling does not need additional authorization to use the default AWS managed key to protect the encrypted volumes in your account.

Contents

- [Overview \(p. 353\)](#)
- [Configure key policies \(p. 354\)](#)
- [Example 1: Key policy sections that allow access to the customer managed key \(p. 354\)](#)
- [Example 2: Key policy sections that allow cross-account access to the customer managed key \(p. 355\)](#)
- [Edit key policies in the AWS KMS console \(p. 357\)](#)

Overview

The following AWS KMS keys can be used for Amazon EBS encryption when Amazon EC2 Auto Scaling launches instances:

- **AWS managed key** — An encryption key in your account that Amazon EBS creates, owns, and manages. This is the default encryption key for a new account. The AWS managed key is used for encryption unless you specify a customer managed key.
- **Customer managed key** — A custom encryption key that you create, own, and manage. For more information, see [Creating keys](#) in the *AWS Key Management Service Developer Guide*.

Note: The key must be symmetric. Amazon EBS does not support asymmetric customer managed keys.

You configure customer managed keys when creating encrypted snapshots or a launch template that specifies encrypted volumes, or enabling encryption by default.

Configure key policies

Your KMS keys must have a key policy that allows Amazon EC2 Auto Scaling to launch instances with Amazon EBS volumes encrypted with a customer managed key.

Use the examples on this page to configure a key policy to give Amazon EC2 Auto Scaling access to your customer managed key. You can modify the customer managed key's key policy either when the key is created or at a later time.

You must, at minimum, add two policy statements to your key policy for it to work with Amazon EC2 Auto Scaling.

- The first statement allows the IAM identity specified in the `Principal` element to use the customer managed key directly. It includes permissions to perform the AWS KMS `Encrypt`, `Decrypt`, `ReEncrypt*`, `GenerateDataKey*`, and `DescribeKey` operations on the key.
- The second statement allows the IAM identity specified in the `Principal` element to use grants to delegate a subset of its own permissions to AWS services that are integrated with AWS KMS or another principal. This allows them to use the key to create encrypted resources on your behalf.

When you add the new policy statements to your key policy, do not change any existing statements in the policy.

For each of the following examples, arguments that must be replaced, such as a key ID or the name of a service-linked role, are shown as *replaceable text in italics*. In most cases, you can replace the name of the service-linked role with the name of an Amazon EC2 Auto Scaling service-linked role. However, when using a launch configuration to launch Spot Instances, use the role named `AWSServiceRoleForEC2Spot`.

For more information, see the following resources:

- To create a key with the AWS CLI, see [create-key](#).
- To update a key policy with the AWS CLI, see [put-key-policy](#).
- To find a key ID and Amazon Resource Name (ARN), see [Finding the key ID and ARN](#) in the *AWS Key Management Service Developer Guide*.
- For information about Amazon EC2 Auto Scaling service-linked roles, see [Service-linked roles for Amazon EC2 Auto Scaling \(p. 332\)](#).
- For more information about Amazon EBS and KMS, [Amazon EBS Encryption](#) in the *Amazon EC2 User Guide for Linux Instances* and the [AWS Key Management Service Developer Guide](#).

Example 1: Key policy sections that allow access to the customer managed key

Add the following two policy statements to the key policy of the customer managed key, replacing the example ARN with the ARN of the appropriate service-linked role that is allowed access to the key. In this example, the policy sections give the service-linked role named `AWSServiceRoleForAutoScaling` permissions to use the customer managed key.

```
{  
  "Sid": "Allow service-linked role use of the customer managed key",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": [  
      "arn:aws:iam::123456789012:role/aws-service-role/  
      autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"  
    ]  
  }  
}
```

```

        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
        ],
        "Resource": "*"
    }
}

```

```

{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
    }
}

```

Example 2: Key policy sections that allow cross-account access to the customer managed key

If you create a customer managed key in a different account than the Auto Scaling group, you must use a grant in combination with the key policy to allow cross-account access to the key.

There are two steps that must be completed in the following order:

1. First, add the following two policy statements to the customer managed key's key policy. Replace the example ARN with the ARN of the other account, making sure to replace **111122223333** with the actual account ID of the AWS account that you want to create the Auto Scaling group in. This allows you to give an IAM user or role in the specified account permission to create a grant for the key using the CLI command that follows. However, this does not by itself give any IAM users or roles access to the key.

```

{
    "Sid": "Allow external account 111122223333 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:root"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}

```

```

        "Resource": "*"
    }

{
    "Sid": "Allow attachment of persistent resources in external account 111122223333",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:root"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*"
}

```

2. Then, from the account that you want to create the Auto Scaling group in, create a grant that delegates the relevant permissions to the appropriate service-linked role. The `Grantee Principal` element of the grant is the ARN of the appropriate service-linked role. The `key-id` is the ARN of the key.

The following is an example `create-grant` CLI command that gives the service-linked role named `AWSServiceRoleForAutoScaling` in account [111122223333](#) permissions to use the customer managed key in account [444455556666](#).

```

aws kms create-grant \
--region us-west-2 \
--key-id arn:aws:kms:us-west-2:44445556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
--grantee-principal arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
--operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey" "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"

```

For this command to succeed, the user making the request must have permissions for the `CreateGrant` action.

The following example IAM policy allows an IAM user or role in account [111122223333](#) to create a grant for the customer managed key in account [444455556666](#).

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455566666",
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "arn:aws:kms:us-west-2:44445556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        }
    ]
}

```

For more information about creating a grant for a KMS key in a different AWS account, see [Grants in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

Important

The service-linked role name specified as the grantee principal must be the name of an existing role. After creating the grant, to ensure that the grant allows Amazon EC2 Auto Scaling to use the specified KMS key, do not delete and recreate the service-linked role.

Edit key policies in the AWS KMS console

The examples in the previous sections show only how to add statements to a key policy, which is just one way of changing a key policy. The easiest way to change a key policy is to use the AWS KMS console's default view for key policies and make an IAM entity (user or role) one of the *key users* for the appropriate key policy. For more information, see [Using the AWS Management Console default view in the AWS Key Management Service Developer Guide](#).

Important

Be cautious. The console's default view policy statements include permissions to perform AWS KMS Revoke operations on the customer managed key. If you give an AWS account access to a customer managed key in your account, and you accidentally revoke the grant that gave them this permission, external users can no longer access their encrypted data or the key that was used to encrypt their data.

Compliance validation for Amazon EC2 Auto Scaling

Third-party auditors assess the security and compliance of Amazon EC2 Auto Scaling as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. 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 AWS services 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 baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

Note

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [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.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

PCI DSS compliance

Amazon EC2 Auto Scaling supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see [PCI DSS Level 1](#).

For information on achieving PCI DSS compliance for your AWS workloads, refer to the following compliance guide:

- [Payment Card Industry Data Security Standard \(PCI DSS\) 3.2.1 on AWS](#)

Resilience in Amazon EC2 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](#).

Related topics

- [Resilience in Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*

Infrastructure security in Amazon EC2 Auto Scaling

As a managed service, Amazon EC2 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 Amazon EC2 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.

Related topics

- [Infrastructure security in Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*

Amazon EC2 Auto Scaling and interface VPC endpoints

You can improve the security posture of your VPC by configuring Amazon EC2 Auto Scaling to use an interface VPC endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that enables

you to privately access Amazon EC2 Auto Scaling APIs by restricting all network traffic between your VPC and Amazon EC2 Auto Scaling to the AWS network. With interface endpoints, you also don't need an internet gateway, a NAT device, or a virtual private gateway.

You are not required to configure AWS PrivateLink, but it's recommended. For more information about AWS PrivateLink and VPC endpoints, see [What is AWS PrivateLink?](#) in the *AWS PrivateLink Guide*.

Topics

- [Create an interface VPC endpoint \(p. 359\)](#)
- [Create a VPC endpoint policy \(p. 359\)](#)

Create an interface VPC endpoint

Create an endpoint for Amazon EC2 Auto Scaling using the following service name:

```
com.amazonaws.region.autoscaling
```

For more information, see [Access an AWS service using an interface VPC endpoint](#) in the *AWS PrivateLink Guide*.

You do not need to change any Amazon EC2 Auto Scaling settings. Amazon EC2 Auto Scaling calls other AWS services using either service endpoints or private interface VPC endpoints, whichever are in use.

Create a VPC endpoint policy

You can attach a policy to your VPC endpoint to control access to the Amazon EC2 Auto Scaling API. The policy specifies:

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

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

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

For more information, see [VPC endpoint policies](#) in the *AWS PrivateLink Guide*.

Troubleshoot Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling provides specific and descriptive errors to help you troubleshoot issues. You can find the error messages in the description of the scaling activities.

Topics

- [Retrieve an error message from scaling activities \(p. 360\)](#)
- [Additional troubleshooting resources \(p. 361\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: EC2 instance launch failures \(p. 362\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: AMI issues \(p. 368\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: Load balancer issues \(p. 369\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: Launch templates \(p. 371\)](#)
- [Troubleshoot Amazon EC2 Auto Scaling: Health checks \(p. 372\)](#)

Retrieve an error message from scaling activities

To retrieve an error message from the description of scaling activities, use the [describe-scaling-activities](#) command. You have a record of scaling activities that dates back 6 weeks. Scaling activities are ordered by start time, with the latest scaling activities listed first.

Note

The scaling activities are also displayed in the activity history in the Amazon EC2 Auto Scaling console on the **Activity** tab for the Auto Scaling group.

To see the scaling activities for a specific Auto Scaling group, use the following command.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

The following is an example response, where `StatusMessage` contains the current status of the activity and `StatusMessage` contains the error message.

```
{  
  "Activities": [  
    {  
      "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",  
      "AutoScalingGroupName": "my-asg",  
      "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e. Status Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed with ABANDON Result",  
      "Cause": "At 2021-01-11T00:35:52Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-01-11T00:35:53Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.",  
      "StartTime": "2021-01-11T00:35:55.542Z",  
      "EndTime": "2021-01-11T01:06:31Z",  
      "Status": "Cancelled",  
      "StatusMessage": "The instance failed to complete the Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f. The instance was abandoned because the user requested to cancel the scaling activity."  
    }  
  ]  
}
```

```

    "StatusMessage": "Instance failed to complete user's Lifecycle Action:  
Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8abaf1 was abandoned: Lifecycle  
Action Completed with ABANDON Result",
        "Progress": 100,
        "Details": "{\"Subnet ID\":\"subnet-5ea0c127\", \"Availability Zone\":\"us-  
west-2b\"...}",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:283179a2-  
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
    ...
}

```

For a description of the fields in the output, see [Activity](#) in the *Amazon EC2 Auto Scaling API Reference*.

To view scaling activities for a deleted group

To view scaling activities after the Auto Scaling group has been deleted, add the `--include-deleted-groups` option to the [describe-scaling-activities](#) command as follows.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-  
deleted-groups
```

The following is an example response, with a scaling activity for a deleted group.

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance. Status Reason: Your Spot request  
price of 0.001 is lower than the minimum required Spot request fulfillment price of  
0.0031. Launching EC2 instance failed.",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup  
constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3. At  
2021-01-13T20:47:27Z an instance was started in response to a difference between desired  
and actual capacity, increasing the capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than the minimum  
required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\", \"Availability Zone\":\"us-  
west-2b\"...}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-  
west-2:123456789012:autoScalingGroup:283179a2-  
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
        ...
    ]
}
```

Additional troubleshooting resources

The following pages provide additional information for troubleshooting issues with Amazon EC2 Auto Scaling.

- Verify a scaling activity for an Auto Scaling group (p. 161)
- View monitoring graphs in the Amazon EC2 Auto Scaling console (p. 267)
- Health checks for Auto Scaling instances (p. 253)
- Considerations and limitations for lifecycle hooks (p. 196)
- Complete a lifecycle action (p. 207)
- Provide network connectivity for your Auto Scaling instances using Amazon VPC (p. 317)
- Temporarily remove instances from your Auto Scaling group (p. 242)
- Disable a scaling policy for an Auto Scaling group (p. 163)
- Suspend and resume a process for an Auto Scaling group (p. 247)
- Control which Auto Scaling instances terminate during scale in (p. 230)
- Delete your Auto Scaling infrastructure (p. 117)
- Quotas for Amazon EC2 Auto Scaling (p. 10)

The following AWS resources can also be of help:

- [Amazon EC2 Auto Scaling topics in the AWS Knowledge Center](#)
- [Amazon EC2 Auto Scaling questions on AWS re:Post](#)
- [Amazon EC2 Auto Scaling posts in the AWS Compute Blog](#)
- [Troubleshooting CloudFormation in the AWS CloudFormation User Guide](#)

Troubleshooting often requires iterative query and discovery by an expert or from a community of helpers. If you continue to experience issues after trying the suggestions in this section, contact AWS Support (in the AWS Management Console, click **Support, Support Center**) or ask a question on [AWS re:Post](#) using the **Amazon EC2 Auto Scaling** tag.

Troubleshoot Amazon EC2 Auto Scaling: EC2 instance launch failures

This page provides information about your EC2 instances that fail to launch, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see [Retrieve an error message from scaling activities \(p. 360\)](#).

When your EC2 instances fail to launch, you might get one or more of the following error messages:

Launch issues

- [The requested configuration is currently not supported. \(p. 363\)](#)
- [The security group <name of the security group> does not exist. Launching EC2 instance failed. \(p. 363\)](#)
- [The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. \(p. 364\)](#)
- [The requested Availability Zone is no longer supported. Please retry your request... \(p. 364\)](#)
- [Your requested instance type \(<instance type>\) is not supported in your requested Availability Zone \(<instance Availability Zone>\)... \(p. 364\)](#)
- [Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735... \(p. 364\)](#)
- [Invalid device name <device name> / Invalid device name upload. Launching EC2 instance failed. \(p. 365\)](#)

- [Value \(<name associated with the instance storage device>\) for parameter virtualName is invalid... \(p. 365\)](#)
- [EBS block device mappings not supported for instance-store AMIs. \(p. 365\)](#)
- [Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. \(p. 366\)](#)
- [Client.InternalError: Client error on launch. \(p. 366\)](#)
- [We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed. \(p. 367\)](#)
- [There is no Spot capacity available that matches your request. Launching EC2 instance failed. \(p. 367\)](#)
- [<number of instances> instance\(s\) are already running. Launching EC2 instance failed. \(p. 367\)](#)

The requested configuration is currently not supported.

- **Cause:** Some options in your launch template or launch configuration might not be compatible with the instance type, or the instance configuration might not be supported in your requested AWS Region or Availability Zones.
- **Solution:**

Try a different instance configuration. To search for an instance type that meets your requirements, see [Finding an Amazon EC2 instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.

For further guidance to resolve this issue, check the following:

- Ensure that you have chosen an AMI that is supported by your instance type. For example, if the instance type uses an Arm-based AWS Graviton processor instead of an Intel Xeon processor, you need an Arm-compatible AMI.
- Test that the instance type is available in your requested Region and Availability Zones. The newest generation instance types might not yet be available in a given Region or Availability Zone. The older generation instance types might not be available in newer Regions and Availability Zones. To search for instance types offered by location (Region or Availability Zone), use the [describe-instance-type-offerings](#) command. For more information, see [Finding an Amazon EC2 instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.
- If you use Dedicated Instances or Dedicated Hosts, ensure that you have chosen an instance type that's supported as a Dedicated Instance or Dedicated Host.

The security group <name of the security group> does not exist. Launching EC2 instance failed.

- **Cause:** The security group specified in your launch template or launch configuration might have been deleted.
- **Solution:**
 1. Use the [describe-security-groups](#) command to get the list of the security groups associated with your account.
 2. From the list, select the security groups to use. To create a security group instead, use the [create-security-group](#) command.
 3. Create a new launch template or launch configuration.
 4. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed.

- **Cause:** The key pair that was used when launching the instance might have been deleted.
- **Solution:**
 1. Use the [describe-key-pairs](#) command to get the list of the key pairs available to you.
 2. From the list, select the key pair to use. To create a key pair instead, use the [create-key-pair](#) command.
 3. Create a new launch template or launch configuration.
 4. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

The requested Availability Zone is no longer supported. Please retry your request...

- **Error message:** The requested Availability Zone is no longer supported. Please retry your request by not specifying an Availability Zone or choosing <list of available Availability Zones>. Launching EC2 instance failed.
- **Cause:** The Availability Zone associated with your Auto Scaling group might not be currently available.
- **Solution:** Update your Auto Scaling group with the recommendations in the error message.

Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)...

- **Error message:** Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>). Please retry your request by not specifying an Availability Zone or choosing <list of Availability Zones that supports the instance type>. Launching EC2 instance failed.
- **Cause:** The instance type you chose might not be currently available in the Availability Zones specified in your Auto Scaling group.
- **Solution:** Update your Auto Scaling group with the recommendations in the error message.

Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735...

- **Cause:** The Spot maximum price in your request is lower than the Spot price for the instance type that you selected.
- **Solution:** Submit a new request with a higher Spot maximum price (possibly the On-Demand price). Previously, the Spot price you paid was based on bidding. Today, you pay the current Spot price. By

setting your maximum price higher, it gives the Amazon EC2 Spot service a better chance of launching and maintaining your required amount of capacity.

Invalid device name <device name> / Invalid device name upload. Launching EC2 instance failed.

- **Cause 1:** The block device mappings in your launch template or launch configuration might contain block device names that are not available or currently not supported.
- **Solution:**
 1. Verify which device names are available for your specific instance configuration. For more details on device naming, see [Device names on Linux instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
 2. Manually create an Amazon EC2 instance that is not part of the Auto Scaling group and investigate the problem. If the block device naming configuration conflicts with the names in the Amazon Machine Image (AMI), the instance will fail during launch. For more information, see [Block device mappings](#) in the *Amazon EC2 User Guide for Linux Instances*.
 3. After you confirm that your instance launched successfully, use the [describe-volumes](#) command to see how the volumes are exposed to the instance.
 4. Create a new launch template or launch configuration using the device name listed in the volume description.
 5. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

Value (<name associated with the instance storage device>) for parameter virtualName is invalid...

- **Error message:** Value (<name associated with the instance storage device>) for parameter virtualName is invalid. Expected format: 'ephemeralNUMBER'. Launching EC2 instance failed.
- **Cause:** The format specified for the virtual name associated with the block device is incorrect.
- **Solution:**
 1. Create a new launch template or launch configuration by specifying the device name in the virtualName parameter. For information about the device name format, see [Device naming on Linux instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
 2. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

EBS block device mappings not supported for instance-store AMIs.

- **Cause:** The block device mappings specified in the launch template or launch configuration are not supported on your instance.
- **Solution:**
 1. Create a new launch template or launch configuration with block device mappings supported by your instance type. For more information, see [Block device mapping](#) in the *Amazon EC2 User Guide for Linux Instances*.

2. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.

- **Cause:** Your cluster placement group contains an invalid instance type.
- **Solution:**
 1. For information about valid instance types supported by the placement groups, see [Placement groups in the Amazon EC2 User Guide for Linux Instances](#).
 2. Follow the instructions detailed in the [Placement groups](#) to create a new placement group.
 3. Alternatively, create a new launch template or launch configuration with the supported instance type.
 4. Update your Auto Scaling group with a new placement group, launch template, or launch configuration using the [update-auto-scaling-group](#) command.

Client.InternalError: Client error on launch.

- **Problem:** Amazon EC2 Auto Scaling tries to launch an instance that has an encrypted EBS volume, but the service-linked role does not have access to the AWS KMS customer managed key used to encrypt it. For more information, see [Required AWS KMS key policy for use with encrypted volumes \(p. 353\)](#).
- **Cause 1:** You need a key policy that gives permission to use the customer managed key to the proper service-linked role.
- **Solution 1:** Allow the service-linked role to use the customer managed key as follows:
 1. Determine which service-linked role to use for this Auto Scaling group.
 2. Update the key policy on the customer managed key and allow the service-linked role to use the customer managed key.
 3. Update the Auto Scaling group to use the service-linked role.

For an example of a key policy that lets the service-linked role use the customer managed key, see [Example 1: Key policy sections that allow access to the customer managed key \(p. 354\)](#).

- **Cause 2:** If the customer managed key and Auto Scaling group are in *different* AWS accounts, you need to configure cross-account access to the customer managed key in order to give permission to use the customer managed key to the proper service-linked role.
- **Solution 2:** Allow the service-linked role in the external account to use the customer managed key in the local account as follows:
 1. Update the key policy on the customer managed key to allow the Auto Scaling group account access to the customer managed key.
 2. Define an IAM user or role in the Auto Scaling group account that can create a grant.
 3. Determine which service-linked role to use for this Auto Scaling group.
 4. Create a grant to the customer managed key with the proper service-linked role as the grantee principal.
 5. Update the Auto Scaling group to use the service-linked role.

For more information, see [Example 2: Key policy sections that allow cross-account access to the customer managed key \(p. 355\)](#).

- **Solution 3:** Use a customer managed key in the same AWS account as the Auto Scaling group.

1. Copy and re-encrypt the snapshot with another customer managed key that belongs to the same account as the Auto Scaling group.
2. Allow the service-linked role to use the new customer managed key. See the steps for Solution 1.

We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed.

- **Error message:** We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>). Our system will be working on provisioning additional capacity. You can currently get <instance type> capacity by not specifying an Availability Zone in your request or choosing <list of Availability Zones that currently supports the instance type>. Launching EC2 instance failed.
- **Cause:** At this time, Amazon EC2 cannot support your instance type in your requested Availability Zone.
- **Solution:**

To resolve the issue, try the following:

- Wait a few minutes and then submit your request again; capacity can shift frequently.
- Submit a new request following the recommendations in the error message.
- Submit a new request with a reduced number of instances (which you can increase at a later stage).

There is no Spot capacity available that matches your request. Launching EC2 instance failed.

- **Cause:** At this time, there isn't enough spare capacity to fulfill your request for Spot Instances.
- **Solution:**

To resolve the issue, try the following:

- Wait a few minutes; capacity can shift frequently. The Spot request continues to automatically make the launch request until capacity becomes available. When capacity becomes available, the Amazon EC2 Spot service fulfills the Spot request.
- Follow the best practice of using a diverse set of instance types so that you are not reliant on one specific instance type. For more information, including a list of best practices for using Spot Instances successfully, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).
- Submit a new request with a reduced number of instances (which you can increase at a later stage).

<number of instances> instance(s) are already running. Launching EC2 instance failed.

- **Cause:** You have reached the limit on the number of instances that you can launch in a Region. When you create your AWS account, we set default limits on the number of instances you can run on a per-Region basis.
- **Solution:**

To resolve the issue, try the following:

- If your current limits aren't adequate for your needs, you can request a quota increase on a per-Region basis. For more information, see [Amazon EC2 service quotas](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Submit a new request with a reduced number of instances (which you can increase at a later stage).

Troubleshoot Amazon EC2 Auto Scaling: AMI issues

This page provides information about the issues associated with your AMIs, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see [Retrieve an error message from scaling activities \(p. 360\)](#).

When your EC2 instances fail to launch due to issues with your AMI, you might get one or more of the following error messages.

AMI issues

- [The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. \(p. 368\)](#)
- [AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. \(p. 368\)](#)
- [Value \(<ami ID>\) for parameter virtualName is invalid. \(p. 368\)](#)
- [The requested instance type's architecture \(i386\) does not match the architecture in the manifest for ami-6622f00f \(x86_64\). Launching EC2 instance failed. \(p. 369\)](#)

The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.

- **Cause:** The AMI might have been deleted after creating the launch template or launch configuration.
- **Solution:**
 1. Create a new launch template or launch configuration using a valid AMI.
 2. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.

- **Cause:** You might have just created your AMI (by taking a snapshot of a running instance or any other way), and it might not be available yet.
- **Solution:** You must wait for your AMI to be available and then create your launch template or launch configuration.

Value (<ami ID>) for parameter virtualName is invalid.

- **Cause:** Incorrect value. The `virtualName` parameter refers to the virtual name associated with the device.

- **Solution:**

1. Create a new launch template or launch configuration by specifying the name of the virtual device of your instance for the `virtualName` parameter.
2. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching EC2 instance failed.

- **Cause:** The architecture of the `InstanceType` mentioned in your launch template or launch configuration does not match the image architecture.

- **Solution:**

1. Create a new launch template or launch configuration using the AMI architecture that matches the architecture of the requested instance type.
2. Update your Auto Scaling group with the new launch template or launch configuration using the [update-auto-scaling-group](#) command.

Troubleshoot Amazon EC2 Auto Scaling: Load balancer issues

This page provides information about issues caused by the load balancer associated with your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see [Retrieve an error message from scaling activities \(p. 360\)](#).

When your EC2 instances fail to launch due to issues with the load balancer associated with your Auto Scaling group, you might get one or more of the following error messages.

Load balancer issues

- [One or more target groups not found. Validating load balancer configuration failed. \(p. 370\)](#)
- [Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed. \(p. 370\)](#)
- [There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. \(p. 370\)](#)
- [EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. \(p. 370\)](#)
- [EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed. \(p. 371\)](#)

Note

You can use VPC Reachability Analyzer to troubleshoot connectivity issues by checking whether instances in your Auto Scaling group are reachable through the load balancer. To learn about the different network misconfiguration issues that are automatically detected by Reachability Analyzer, see [VPC Reachability Analyzer explanation codes](#) in the *VPC Reachability Analyzer User Guide*.

One or more target groups not found. Validating load balancer configuration failed.

Problem: When your Auto Scaling group launches instances, Amazon EC2 Auto Scaling tries to validate that the Elastic Load Balancing resources that are associated with the Auto Scaling group exist. When a target group cannot be found, the scaling activity fails, and you get the One or more target groups not found. Validating load balancer configuration failed. error.

Cause 1: A target group attached to your Auto Scaling group has been deleted.

Solution 1: You can either create a new Auto Scaling group without the target group or remove the unused target group from the Auto Scaling group by using the Amazon EC2 Auto Scaling console or the [detach-load-balancer-target-groups](#) command.

Cause 2: The target group exists, but there was an issue trying to specify the target group ARN when creating the Auto Scaling group. Resources are not created in the right order.

Solution 2: Create a new Auto Scaling group and specify the target group at the end.

Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed.

Problem: When your Auto Scaling group launches instances, Amazon EC2 Auto Scaling tries to validate that the Elastic Load Balancing resources that are associated with the Auto Scaling group exist. When a Classic Load Balancer cannot be found, the scaling activity fails, and you get the Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed. error.

Cause 1: The Classic Load Balancer has been deleted.

Solution 1: You can either create a new Auto Scaling group without the load balancer or remove the unused load balancer from the Auto Scaling group by using the Amazon EC2 Auto Scaling console or the [detach-load-balancers](#) command.

Cause 2: The Classic Load Balancer exists, but there was an issue trying to specify the load balancer name when creating the Auto Scaling group. Resources are not created in the right order.

Solution 2: Create a new Auto Scaling group and specify the load balancer name at the end.

There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.

Cause: The specified load balancer might have been deleted.

Solution: You can either create a new load balancer and then create a new Auto Scaling group or create a new Auto Scaling group without the load balancer.

EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.

Cause: The specified instance does not exist in the VPC.

Solution: You can either delete your load balancer associated with the instance or create a new Auto Scaling group.

EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed.

Cause: The load balancer is in EC2-Classic but the Auto Scaling group is in a VPC.

Solution: Ensure that the load balancer and the Auto Scaling group are in the same network (EC2-Classic or a VPC).

Troubleshoot Amazon EC2 Auto Scaling: Launch templates

Use the following information to help you diagnose and fix common issues that you might encounter when trying to specify a launch template for your Auto Scaling group.

Can't launch instances

If you are unable to launch any instances with an already specified launch template, check the following for general troubleshooting: [Troubleshoot Amazon EC2 Auto Scaling: EC2 instance launch failures \(p. 362\)](#).

You must use a valid fully-formed launch template (invalid value)

Problem: When you try to specify a launch template for an Auto Scaling group, you get the You must use a valid fully-formed launch template error. You might encounter this error because the values in the launch template are only validated when an Auto Scaling group that is using the launch template is created or updated

Cause 1: If you receive a You must use a valid fully-formed launch template error, then there are issues that cause Amazon EC2 Auto Scaling to consider something about the launch template not valid. This is a generic error that can have several different causes.

Solution 1: Try the following steps to troubleshoot:

1. Pay attention to the second part of the error message to find more information. Following the You must use a valid fully-formed launch template error, see the more specific error message that identifies the issue that you will need to address.
2. If you are unable to find the cause, test your launch template with the `RunInstances` command. Use the `--dry-run` option, as shown in the following example. This lets you reproduce the issue and can provide insights about its cause.

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --dry-run
```

3. If a value is not valid, verify that the specified resource exists and that it's correct. For example, when you specify an Amazon EC2 key pair, the resource must exist in your account and in the Region in which you are creating or updating your Auto Scaling group.
4. If expected information is missing, verify your settings and adjust the launch template as needed.

5. After making your changes, re-run the [RunInstances](#) command with the `--dry-run` option to verify that your launch template uses valid values.

For more information, see [Create a launch template for an Auto Scaling group \(p. 22\)](#).

You are not authorized to use launch template (insufficient IAM permissions)

Problem: When you try to specify a launch template for an Auto Scaling group, you get the You are not authorized to use launch template error.

Cause 1: If you are attempting to use a launch template, and the IAM credentials that you are using do not have sufficient permissions, you receive an error that you're not authorized to use the launch template.

Solution 1: Verify that the IAM credentials that you are using to make the request has permissions to call the EC2 API actions you need, including the `ec2:RunInstances` action. If you specified any tags in your launch template, you must also have permission to use the `ec2:CreateTags` action.

Solution 2: Verify that the IAM credentials that you are using to make the request is assigned the `AmazonEC2FullAccess` policy. This AWS managed policy grants full access to all Amazon EC2 resources and related services, including Amazon EC2 Auto Scaling, CloudWatch, and Elastic Load Balancing.

Cause 2: If you are attempting to use a launch template that specifies an instance profile, you must have IAM permission to pass the IAM role that is associated with the instance profile.

Solution 3: Verify that the IAM credentials that you are using to make the request has the correct permissions to pass the specified role to the Amazon EC2 Auto Scaling service. For more information, see [IAM role for applications that run on Amazon EC2 instances \(p. 351\)](#). For further troubleshooting topics related to instance profiles, see [Troubleshooting Amazon EC2 and IAM](#) in the *IAM User Guide*.

For more information about setting up IAM permissions for launch templates, see [Launch template support \(p. 347\)](#).

Troubleshoot Amazon EC2 Auto Scaling: Health checks

This page provides information about your EC2 instances that terminate due to a health check. It describes potential causes, and the steps that you can take to resolve the issues.

To retrieve an error message, see [Retrieve an error message from scaling activities \(p. 360\)](#).

Health check issues

- An instance was taken out of service in response to an EC2 instance status check failure (p. 373)
- An instance was taken out of service in response to an EC2 scheduled reboot (p. 373)
- An instance was taken out of service in response to an EC2 health check that indicated it had been terminated or stopped (p. 374)
- An instance was taken out of service in response to an ELB system health check failure (p. 374)

Note

You can be notified when Amazon EC2 Auto Scaling terminates the instances in your Auto Scaling group, including when the cause of instance termination is not the result of a scaling

activity. For more information, see [Get Amazon SNS notifications when your Auto Scaling group scales \(p. 271\)](#).

The sections that follow describe the most common health check errors and causes that you'll encounter. If you have a different issue, see the following AWS Knowledge Center articles for additional troubleshooting help:

- [Why did Amazon EC2 Auto Scaling terminate an instance?](#)
- [Why didn't Amazon EC2 Auto Scaling terminate an unhealthy instance?](#)

An instance was taken out of service in response to an EC2 instance status check failure

Problem: Auto Scaling instances fail the Amazon EC2 status checks.

Cause 1: If there are issues that cause Amazon EC2 to consider the instances in your Auto Scaling group impaired, Amazon EC2 Auto Scaling automatically replaces the impaired instances as part of its health check. Status checks are built into Amazon EC2, so they cannot be disabled or deleted. When an instance status check fails, you typically must address the problem yourself by making instance configuration changes until your application is no longer exhibiting any problems.

Solution 1: To address this issue, follow these steps:

1. Manually create an Amazon EC2 instance that is not part of the Auto Scaling group and investigate the problem. For general help with investigating impaired instances, see [Troubleshoot instances with failed status checks in the Amazon EC2 User Guide for Linux Instances](#) and [Troubleshooting Windows Instances in the Amazon EC2 User Guide for Windows Instances](#).
2. After you confirm that your instance launched successfully and is healthy, deploy a new, error-free instance configuration to the Auto Scaling group.
3. Delete the instance that you created to avoid ongoing charges to your AWS account.

Cause 2: There is a mismatch between the health check grace period and the instance startup time.

Solution 2: Edit the health check grace period for your Auto Scaling group to an appropriate time period for your application. Instances launched in an Auto Scaling group require sufficient warm-up time (grace period) to prevent early termination due to a health check replacement. For more information, see [Set the health check grace period for an Auto Scaling group \(p. 258\)](#).

An instance was taken out of service in response to an EC2 scheduled reboot

Problem: Auto Scaling instances are replaced when a scheduled event indicates a problem with the instance.

Cause: Amazon EC2 Auto Scaling replaces instances with a future scheduled maintenance or retirement event.

Solution: These events do not occur frequently. If you need something to happen on the instance that is terminating, or on the instance that is starting up, you can use lifecycle hooks. These hooks allow you to perform a custom action as Amazon EC2 Auto Scaling launches or terminates instances. For more information, see [Amazon EC2 Auto Scaling lifecycle hooks \(p. 195\)](#).

If you do not want instances to be replaced due to a scheduled event, you can suspend the health check process for an Auto Scaling group. For more information, see [Suspend and resume a process for an Auto Scaling group \(p. 247\)](#).

An instance was taken out of service in response to an EC2 health check that indicated it had been terminated or stopped

Problem: Auto Scaling instances that have been stopped, rebooted, or terminated are replaced.

Cause 1: A user manually stopped, rebooted, or terminated the instance.

Solution 1: If a health check fails because a user manually stopped, rebooted, or terminated the instance, this is due to how Amazon EC2 Auto Scaling health checks work. The instance must be healthy and reachable. If you need to reboot the instances in your Auto Scaling group, we recommend that you put the instances on standby first. For more information, see [Temporarily remove instances from your Auto Scaling group \(p. 242\)](#).

Note that when you terminate instances manually, termination lifecycle hooks and Elastic Load Balancing deregistration (and connection draining) must be completed before the instance is actually terminated.

Cause 2: Amazon EC2 Auto Scaling attempts to replace Spot Instances after the Amazon EC2 Spot service interrupts the instances, because the Spot price increases above your maximum price or capacity is no longer available.

Solution 2: There is no guarantee that a Spot Instance exists to fulfill the request at any given point in time. However, you can try the following:

- Use a higher Spot maximum price (possibly the On-Demand price). By setting your maximum price higher, it gives the Amazon EC2 Spot service a better chance of launching and maintaining your required amount of capacity.
- Increase the number of different capacity pools that you can launch instances from by running multiple instance types in multiple Availability Zones. For more information, see [Auto Scaling groups with multiple instance types and purchase options \(p. 52\)](#).
- If you use multiple instance types, consider enabling the Capacity Rebalancing feature. This is useful if you want the Amazon EC2 Spot service to attempt to launch a new Spot Instance before a running instance is terminated. For more information, see [Use Capacity Rebalancing to handle Amazon EC2 Spot interruptions \(p. 275\)](#).

An instance was taken out of service in response to an ELB system health check failure

Problem: Auto Scaling instances might pass the EC2 status checks. But they might fail the Elastic Load Balancing health checks for the target groups or Classic Load Balancers with which the Auto Scaling group is registered.

Cause: If your Auto Scaling group relies on health checks provided by Elastic Load Balancing, Amazon EC2 Auto Scaling determines the health status of your instances by checking the results of both the EC2 status checks and the Elastic Load Balancing health checks. The load balancer performs health checks by sending a request to each instance and waiting for the correct response, or by establishing a connection with the instance. An instance might fail the Elastic Load Balancing health check because an application running on the instance has issues that cause the load balancer to consider the instance out of service. For more information, see [Add Elastic Load Balancing health checks to an Auto Scaling group \(p. 291\)](#).

Solution 1: To pass the Elastic Load Balancing health checks:

- Make note of the success codes that the load balancer is expecting, and verify that your application is configured correctly to return these codes on success.
- Verify that the security groups for your load balancer and Auto Scaling group are correctly configured.
- Verify that the health check settings of your target groups are correctly configured. You define health check settings for your load balancer per target group.
- Consider adding a launch lifecycle hook to the Auto Scaling group to ensure that the applications on the instances are ready to accept traffic before they are registered to the load balancer at the end of the lifecycle hook.
- Set the health check grace period for your Auto Scaling group to a long enough time period to support the number of consecutive successful health checks required before Elastic Load Balancing considers a newly launched instance healthy.
- Verify that the load balancer is configured in the same Availability Zones as your Auto Scaling group.

For more information, see the following topics:

- [Health checks for your target groups](#) in the *User Guide for Application Load Balancers*
- [Health checks for your target groups](#) in the *User Guide for Network Load Balancers*
- [Health checks for your target groups](#) in the *User Guide for Gateway Load Balancers*
- [Configure health checks for your Classic Load Balancer](#) in the *User Guide for Classic Load Balancers*

Solution 2: Update the Auto Scaling group to disable Elastic Load Balancing health checks.

Amazon EC2 Auto Scaling resources

The following related resources can help you as you work with this service.

- [Amazon EC2: How to connect to your Linux instance](#) – Learn how to connect to the Linux instances that you launch.
- [Amazon EC2: How to connect to your Windows instance](#) – Learn how to connect to the Windows instances that you launch.
- [CloudWatch: Creating a billing alarm to monitor your estimated AWS charges](#) – Learn how to monitor your estimated charges using CloudWatch.

The following additional resources are available to help you learn more about Amazon Web Services.

- [Classes & Workshops](#) – Links to role-based and specialty courses, in addition to self-paced labs to help sharpen your AWS skills and gain practical experience.
- [AWS Developer Center](#) – Explore tutorials, download tools, and learn about AWS developer events.
- [AWS Developer Tools](#) – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- [Getting Started Resource Center](#) – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- [Hands-On Tutorials](#) – Follow step-by-step tutorials to launch your first application on AWS.
- [AWS Whitepapers](#) – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- [AWS Support Center](#) – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- [AWS Support](#) – The primary webpage for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- [Contact Us](#) – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- [AWS Site Terms](#) – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Document history

The following table describes important additions to the Amazon EC2 Auto Scaling documentation, beginning in July 2018. For notification about updates to this documentation, you can subscribe to the RSS feed.

Change	Description	Date
Support for predictive scaling in Asia Pacific (Jakarta) Region (p. 377)	You can now create predictive scaling policies in Asia Pacific (Jakarta) Region. For more information, see Predictive scaling for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	October 13, 2022
Support for custom metrics for predictive scaling in the console (p. 377)	You can now use custom metrics when creating predictive scaling policies from the Amazon EC2 Auto Scaling console. For more information, see Predictive scaling for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	October 13, 2022
CloudWatch monitoring for predictive scaling metrics (p. 377)	You can now access monitoring data for predictive scaling using CloudWatch. This lets you use metric math to create new time series that display the accuracy of forecast data. For more information, see Monitor predictive scaling metrics with CloudWatch in the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 7, 2022
Support for predictive scaling in Asia Pacific (Osaka) Region (p. 377)	You can now create predictive scaling policies in Asia Pacific (Osaka) Region. For more information, see Predictive scaling for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 6, 2022
Warm pool hibernation supported in additional Regions (p. 377)	You can now hibernate instances in a warm pool in four additional Regions: Africa (Cape Town), Asia Pacific (Jakarta), Asia Pacific (Osaka), and Europe (Milan). For more information about warm pools, see Warm pools for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 5, 2022

Update to health checks (p. 377)	When performing health checks, Amazon EC2 Auto Scaling now helps you minimize any downtime that can occur because of temporary issues or misconfigured health checks. For more information, see How Amazon EC2 Auto Scaling minimizes downtime in the Amazon EC2 Auto Scaling User Guide .	May 21, 2022
Default instance warmup (p. 377)	You can now unify all the warmup and cooldown settings for an Auto Scaling group and optimize the performance of scaling policies that scale continuously by enabling default instance warmup. For more information, see Set the default instance warmup for an Auto Scaling group in the <i>Amazon EC2 Auto Scaling User Guide</i> .	April 19, 2022
Guide changes (p. 377)	A new chapter about integrating with other AWS services has been added to the guide. For more information, see AWS services integrated with Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 29, 2022
Update to IAM service-linked role permissions (p. 377)	The <code>AutoScalingServiceRolePolicy</code> policy now grants additional read permissions to Amazon EC2 Auto Scaling. For more information, see AWS managed policies for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 28, 2022
Instance metadata provides target lifecycle state (p. 377)	You can retrieve an Auto Scaling instance's target lifecycle state from the instance metadata. For more information, see Retrieve the target lifecycle state through instance metadata in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 24, 2022

Support for new warm pool functionality (p. 377)	You can now hibernate instances in a warm pool to stop instances without deleting their memory contents (RAM). You can now also return instances to the warm pool on scale in, instead of always terminating instance capacity that you will need later. For more information, see Warm pools for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 24, 2022
Guide changes (p. 377)	The Amazon EC2 Auto Scaling console has been updated with additional options to help you start an instance refresh with skip matching enabled and a desired configuration specified. For more information, see Start or cancel an instance refresh (console) in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 3, 2022
Custom metrics for predictive scaling policies (p. 377)	You can now choose whether to use custom metrics when you create predictive scaling policies. You can also use metric math to further customize the metrics that you include in your policy. For more information, see Advanced predictive scaling policy configurations using custom metrics .	November 24, 2021
New On-Demand allocation strategy (p. 377)	You can now choose whether to launch On-Demand Instances based on price (lowest priced instance types first) when you create an Auto Scaling group that uses a mixed instances policy. For more information, see Allocation strategies in the <i>Amazon EC2 Auto Scaling User Guide</i> .	October 27, 2021

Attribute-based instance type selection (p. 377)	Amazon EC2 Auto Scaling adds support for attribute-based instance type selection. Instead of manually choosing instance types, you can express your instance requirements as a set of attributes, such as vCPU, memory, and storage. For more information, see Creating an Auto Scaling group using attribute-based instance type selection in the <i>Amazon EC2 Auto Scaling User Guide</i> .	October 27, 2021
Support for filtering groups by tags (p. 377)	You can now filter your Auto Scaling groups using tag filters when you retrieve information about your Auto Scaling groups using the <code>describe-auto-scaling-groups</code> command. For more information, see Use tags to filter Auto Scaling groups in the <i>Amazon EC2 Auto Scaling User Guide</i> .	October 14, 2021
Guide changes (p. 377)	The Amazon EC2 Auto Scaling console has been updated to help you create custom termination policies with AWS Lambda. The console documentation has been revised accordingly. For more information, see Using different termination policies (console) .	October 14, 2021
Support for copying launch configurations to launch templates (p. 377)	You can now copy all launch configurations in an AWS Region to new launch templates from the Amazon EC2 Auto Scaling console. For more information, see Copy launch configurations to launch templates in the <i>Amazon EC2 Auto Scaling User Guide</i> .	August 9, 2021

Expands instance refresh functionality (p. 377)	You can now include updates, such as a new version of a launch template, when replacing instances by adding your desired configuration to the <code>start-instance-refresh</code> command. You can also skip replacing instances that already have your desired configuration by enabling skip matching. For more information, see Replacing Auto Scaling instances based on an instance refresh in the <i>Amazon EC2 Auto Scaling User Guide</i> .	August 5, 2021
Support for custom termination policies (p. 377)	You can now create custom termination policies with AWS Lambda. For more information, see Creating a custom termination policy with Lambda . The documentation for specifying termination policies has been updated accordingly.	July 29, 2021
Guide changes (p. 377)	The Amazon EC2 Auto Scaling console has been updated and enhanced with additional features to help you create scheduled actions with a time zone specified. The documentation for Scheduled scaling has been revised accordingly.	June 3, 2021
gp3 volumes in launch configurations (p. 377)	You can now specify gp3 volumes in the block device mappings for launch configurations.	June 2, 2021
Support for predictive scaling (p. 377)	You can now use predictive scaling to proactively scale your Amazon EC2 Auto Scaling groups using a scaling policy. For more information, see Predictive scaling for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> . With this update, the <code>AutoScalingServiceRolePolicy</code> managed policy now includes permission to call the <code>cloudwatch:GetMetricData</code> API action.	May 19, 2021

Guide changes (p. 377)	You can now access example templates for lifecycle hooks from GitHub. For more information, see Amazon EC2 Auto Scaling lifecycle hooks in the <i>Amazon EC2 Auto Scaling User Guide</i> .	April 9, 2021
Support for warm pools (p. 377)	You can now balance performance (minimize cold starts) and cost (stop over-provisioning instance capacity) for applications with long first boot times by adding warm pools to Auto Scaling groups. For more information, see Warm pools for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	April 8, 2021
Support for checkpoints (p. 377)	You can now add checkpoints to an instance refresh to replace instances in phases and perform verifications on your instances at specific points. For more information, see Adding checkpoints to an instance refresh in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 18, 2021
Guide changes (p. 377)	Improved documentation for using EventBridge with Amazon EC2 Auto Scaling events and lifecycle hooks. For more information, see Using Amazon EC2 Auto Scaling with EventBridge and Tutorial: Configure a lifecycle hook that invokes a Lambda function in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 18, 2021
Support for local time zones (p. 377)	You can now create recurring scheduled actions in the local time zone by adding the <code>--time-zone</code> option to the <code>put-scheduled-update-group-action</code> command. If your time zone observes Daylight Saving Time (DST), the recurring action automatically adjusts for Daylight Saving Time. For more information, see Scheduled scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 9, 2021

Expands functionality for mixed instances policies (p. 377)	You can now prioritize instance types for your Spot capacity when you use a mixed instances policy. Amazon EC2 Auto Scaling attempts to fulfill priorities on a best-effort basis but optimizes for capacity first. For more information, see Auto Scaling groups with multiple instance types and purchase options in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 8, 2021
Scaling activities for deleted groups (p. 377)	You can now view scaling activities for deleted Auto Scaling groups by adding the <code>--include-deleted-groups</code> option to the describe-scaling-activities command. For more information, see Troubleshooting Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 23, 2021
Console improvements (p. 377)	You can now create and attach an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console. For more information, see Create and attach a new Application Load Balancer or Network Load Balancer (console) in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 24, 2020
Multiple network interfaces (p. 377)	You can now configure a launch template for an Auto Scaling group that specifies multiple network interfaces. For more information, see Network interfaces in a VPC .	November 23, 2020
Multiple launch templates (p. 377)	Multiple launch templates can now be used with Auto Scaling groups. For more information, see Specifying a different launch template for an instance type in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 19, 2020

Gateway Load Balancers (p. 377)	Updated guide to show how to attach a Gateway Load Balancer to an Auto Scaling group to ensure that appliance instances launched by Amazon EC2 Auto Scaling are automatically registered and deregistered from the load balancer. For more information, see Elastic Load Balancing types and Attaching a load balancer to your Auto Scaling group in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 10, 2020
Maximum instance lifetime (p. 377)	You can now reduce the maximum instance lifetime to one day (86,400 seconds). For more information, see Replacing Auto Scaling instances based on maximum instance lifetime in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 9, 2020
Capacity Rebalancing (p. 377)	You can configure your Auto Scaling group to launch a replacement Spot Instance when Amazon EC2 emits a rebalance recommendation. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 4, 2020
Instance metadata service version 2 (p. 377)	You can require the use of Instance Metadata Service Version 2, which is a session-oriented method for requesting instance metadata, when using launch configurations. For more information, see Configuring the instance metadata options in the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 28, 2020
Guide changes (p. 377)	Various improvements and new console procedures in the Controlling which Auto Scaling instances terminate during scale in , Monitoring your Auto Scaling instances and groups , Launch templates , and Launch configurations sections of the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 28, 2020

Instance refresh (p. 377)	Start an instance refresh to update all instances in your Auto Scaling group when you make a configuration change. For more information, see Replacing Auto Scaling instances based on an instance refresh in the <i>Amazon EC2 Auto Scaling User Guide</i> .	June 16, 2020
Guide changes (p. 377)	Various improvements in the Replacing Auto Scaling instances based on maximum instance lifetime , Auto Scaling groups with multiple instance types and purchase options , Scaling based on Amazon SQS , and Tagging Auto Scaling groups and instances sections of the <i>Amazon EC2 Auto Scaling User Guide</i> .	May 6, 2020
Guide changes (p. 377)	Various improvements to IAM documentation. For more information, see Launch template support and Amazon EC2 Auto Scaling identity-based policy examples in the <i>Amazon EC2 Auto Scaling User Guide</i> .	March 4, 2020
Disable scaling policies (p. 377)	You can now disable and re-enable scaling policies. This feature allows you to temporarily disable a scaling policy while preserving the configuration details so that you can enable the policy again later. For more information, see Disabling a scaling policy for an Auto Scaling group in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 18, 2020
Add notification functionality (p. 377)	Amazon EC2 Auto Scaling now sends events to your AWS Health Dashboard when your Auto Scaling groups cannot scale out due to a missing security group or launch template. For more information, see AWS Health Dashboard notifications for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 12, 2020

Guide changes (p. 377)	Various improvements and corrections in the How Amazon EC2 Auto Scaling works with IAM , Amazon EC2 Auto Scaling identity-based policy examples , Required CMK key policy for use with encrypted volumes , and Monitoring your Auto Scaling instances and groups sections of the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 10, 2020
Guide changes (p. 377)	Improved documentation for Auto Scaling groups that use instance weighting. Learn how to use scaling policies when using "capacity units" to measure desired capacity. For more information, see How scaling policies work and Scaling adjustment types in the <i>Amazon EC2 Auto Scaling User Guide</i> .	February 6, 2020
New "Security" chapter (p. 377)	A new Security chapter in the <i>Amazon EC2 Auto Scaling User Guide</i> helps you understand how to apply the shared responsibility model when using Amazon EC2 Auto Scaling. As part of this update, the user guide chapter "Controlling Access to Your Amazon EC2 Auto Scaling Resources" has been replaced by a new, more useful section, Identity and access management for Amazon EC2 Auto Scaling .	February 4, 2020
Recommendations for instance types (p. 377)	AWS Compute Optimizer provides Amazon EC2 instance recommendations to help you improve performance, save money, or both. For more information, see Getting recommendations for an instance type in the <i>Amazon EC2 Auto Scaling User Guide</i> .	December 3, 2019

Dedicated Hosts and host resource groups (p. 377)	Updated guide to show how to create a launch template that specifies a host resource group. This allows you to create an Auto Scaling group with a launch template that specifies a BYOL AMI to use on Dedicated Hosts. For more information, see Creating a launch template for an Auto Scaling group in the <i>Amazon EC2 Auto Scaling User Guide</i> .	December 3, 2019
Support for Amazon VPC endpoints (p. 377)	You can now establish a private connection between your VPC and Amazon EC2 Auto Scaling. For more information, see Amazon EC2 Auto Scaling and interface VPC endpoints in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 22, 2019
Maximum instance lifetime (p. 377)	You can now replace instances automatically by specifying the maximum length of time that an instance can be in service. If any instances are approaching this limit, Amazon EC2 Auto Scaling gradually replaces them. For more information, see Replacing Auto Scaling instances based on maximum instance lifetime in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 19, 2019
Instance weighting (p. 377)	For Auto Scaling groups with multiple instance types, you can now optionally specify the number of capacity units that each instance type contributes to the capacity of the group. For more information, see Instance weighting for Amazon EC2 Auto Scaling in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 19, 2019
Minimum number of instance types (p. 377)	You no longer have to specify additional instance types for groups of Spot, On-Demand, and Reserved Instances. For all Auto Scaling groups, the minimum is now one instance type. For more information, see Auto Scaling groups with multiple instance types and purchase options in the <i>Amazon EC2 Auto Scaling User Guide</i> .	September 16, 2019

Support for new Spot allocation strategy (p. 377)	Amazon EC2 Auto Scaling now supports a new Spot allocation strategy "capacity-optimized" that fulfills your request using Spot Instance pools that are optimally chosen based on the available Spot capacity. For more information, see Auto Scaling groups with multiple instance types and purchase options in the <i>Amazon EC2 Auto Scaling User Guide</i> .	August 12, 2019
Guide changes (p. 377)	Improved Amazon EC2 Auto Scaling documentation in the Service-linked roles and Required CMK key policy for use with encrypted volumes topics.	August 1, 2019
Support for tagging enhancement (p. 377)	Amazon EC2 Auto Scaling now adds tags to Amazon EC2 instances as part of the same API call that launches the instances. For more information, see Tagging Auto Scaling groups and instances .	July 26, 2019
Guide changes (p. 377)	Improved Amazon EC2 Auto Scaling documentation in the Suspending and resuming scaling processes topic. Updated Customer managed policy examples to include an example policy that allows users to pass only specific custom suffix service-linked roles to Amazon EC2 Auto Scaling.	June 13, 2019
Support for new Amazon EBS feature (p. 377)	Added support for new Amazon EBS feature in the launch template topic. Change the encryption state of an EBS volume while restoring from a snapshot. For more information, see Creating a launch template for an Auto Scaling group in the <i>Amazon EC2 Auto Scaling User Guide</i> .	May 13, 2019

Guide changes (p. 377)	Improved Amazon EC2 Auto Scaling documentation in the following sections: Controlling which Auto Scaling instances terminate during scale in , Auto Scaling groups , Auto Scaling groups with multiple instance types and purchase options , and Dynamic scaling for Amazon EC2 Auto Scaling .	March 12, 2019
Support for combining instance types and purchase options (p. 377)	Provision and automatically scale instances across purchase options (Spot, On-Demand, and Reserved Instances) and instance types within a single Auto Scaling group. For more information, see Auto Scaling groups with multiple instance types and purchase options in the <i>Amazon EC2 Auto Scaling User Guide</i> .	November 13, 2018
Updated topic for scaling based on Amazon SQS (p. 377)	Updated guide to explain how you can use custom metrics to scale an Auto Scaling group in response to changing demand from an Amazon SQS queue. For more information, see Scaling based on Amazon SQS in the <i>Amazon EC2 Auto Scaling User Guide</i> .	July 26, 2018

The following table describes important changes to the Amazon EC2 Auto Scaling documentation before July 2018.

Feature	Description	Release date
Support for target tracking scaling policies	Set up dynamic scaling for your application in just a few steps. For more information, see Target tracking scaling policies for Amazon EC2 Auto Scaling .	12 July 2017
Support for resource-level permissions	Create IAM policies to control access at the resource level. For more information, see Controlling access to your Amazon EC2 Auto Scaling resources .	15 May 2017
Monitoring improvements	Auto Scaling group metrics no longer require that you enable detailed monitoring. You can now enable group metrics collection and view metrics graphs from the Monitoring tab in the console. For more information, see Monitoring your Auto Scaling groups and instances using Amazon CloudWatch .	18 August 2016
Support for Application Load Balancers	Attach one or more target groups to a new or existing Auto Scaling group. For more information, see Attaching a load balancer to your Auto Scaling group .	11 August 2016

Feature	Description	Release date
Events for lifecycle hooks	Amazon EC2 Auto Scaling sends events to EventBridge when it calls lifecycle hooks. For more information, see Getting EventBridge when your Auto Scaling group scales .	24 February 2016
Instance protection	Prevent Amazon EC2 Auto Scaling from selecting specific instances for termination when scaling in. For more information, see Instance protection .	07 December 2015
Step scaling policies	Create a scaling policy that enables you to scale based on the size of the alarm breach. For more information, see Scaling policy types .	06 July 2015
Update load balancer	Attach a load balancer to or detach a load balancer from an existing Auto Scaling group. For more information, see Attaching a load balancer to your Auto Scaling group .	11 June 2015
Support for ClassicLink	Link EC2-Classic instances in your Auto Scaling group to a VPC, enabling communication between these linked EC2-Classic instances and instances in the VPC using private IP addresses. For more information, see Linking EC2-Classic instances to a VPC .	19 January 2015
Lifecycle hooks	Hold your newly launched or terminating instances in a pending state while you perform actions on them. For more information, see Amazon EC2 Auto Scaling lifecycle hooks .	30 July 2014
Detach instances	Detach instances from an Auto Scaling group. For more information, see Detach EC2 instances from your Auto Scaling group .	30 July 2014
Put instances into a Standby state	Put instances that are in an <code>InService</code> state into a <code>Standby</code> state. For more information, see Temporarily removing instances from your Auto Scaling group .	30 July 2014
Manage tags	Manage your Auto Scaling groups using the AWS Management Console. For more information, see Tagging Auto Scaling groups and instances .	01 May 2014
Support for Dedicated Instances	Launch Dedicated Instances by specifying a placement tenancy attribute when you create a launch configuration. For more information, see Instance placement tenancy .	23 April 2014
Create a group or launch configuration from an EC2 instance	Create an Auto Scaling group or a launch configuration using an EC2 instance. For information about creating a launch configuration using an EC2 instance, see Creating a launch configuration using an EC2 instance . For information about creating an Auto Scaling group using an EC2 instance, see Creating an Auto Scaling group using an EC2 instance .	02 January 2014
Attach instances	Enable automatic scaling for an EC2 instance by attaching the instance to an existing Auto Scaling group. For more information, see Attach EC2 instances to your Auto Scaling group .	02 January 2014
View account limits	View the limits on Auto Scaling resources for your account. For more information, see Auto Scaling limits .	02 January 2014

Feature	Description	Release date
Console support for Amazon EC2 Auto Scaling	Access Amazon EC2 Auto Scaling using the AWS Management Console. For more information, see Getting started with Amazon EC2 Auto Scaling .	10 December 2013
Assign a public IP address	Assign a public IP address to an instance launched into a VPC. For more information, see Launching Auto Scaling instances in a VPC .	19 September 2013
Instance termination policy	Specify an instance termination policy for Amazon EC2 Auto Scaling to use when terminating EC2 instances. For more information, see Controlling which Auto Scaling instances terminate during scale in .	17 September 2012
Support for IAM roles	Launch EC2 instances with an IAM instance profile. You can use this feature to assign IAM roles to your instances, allowing your applications to access other Amazon Web Services securely. For more information, see Launch Auto Scaling instances with an IAM role .	11 June 2012
Support for Spot Instances	Launch Spot Instances with a launch configuration. For more information, see Requesting Spot Instances for fault-tolerant and flexible applications .	7 June 2012
Tag groups and instances	Tag Auto Scaling groups and specify that the tag also applies to EC2 instances launched after the tag was created. For more information, see Tagging Auto Scaling groups and instances .	26 January 2012
Support for Amazon SNS	<p>Use Amazon SNS to receive notifications whenever Amazon EC2 Auto Scaling launches or terminates EC2 instances. For more information, see Getting SNS notifications when your Auto Scaling group scales.</p> <p>Amazon EC2 Auto Scaling also added the following new features:</p> <ul style="list-style-type: none"> The ability to set up recurring scaling activities using cron syntax. For more information, see the PutScheduledUpdateGroupAction API operation. A new configuration setting that allows you to scale out without adding the launched instance to the load balancer (LoadBalancer). For more information, see the ProcessType API data type. The ForceDelete flag in the DeleteAutoScalingGroup operation that tells Amazon EC2 Auto Scaling to delete the Auto Scaling group with the instances associated to it without waiting for the instances to be terminated first. For more information, see the DeleteAutoScalingGroup API operation. 	20 July 2011
Scheduled scaling actions	Added support for scheduled scaling actions. For more information, see Scheduled scaling for Amazon EC2 Auto Scaling .	2 December 2010
Support for Amazon VPC	Added support for Amazon VPC. For more information, see Launching Auto Scaling instances in a VPC .	2 December 2010

Feature	Description	Release date
Support for HPC clusters	Added support for high performance computing (HPC) clusters.	2 December 2010
Support for health checks	Added support for using Elastic Load Balancing health checks with Amazon EC2 Auto Scaling-managed EC2 instances. For more information, see Using ELB health checks with Auto Scaling .	2 December 2010
Support for CloudWatch alarms	Removed the older trigger mechanism and redesigned Amazon EC2 Auto Scaling to use the CloudWatch alarm feature. For more information, see Dynamic scaling for Amazon EC2 Auto Scaling .	2 December 2010
Suspend and resume scaling	Added support to suspend and resume scaling processes.	2 December 2010
Support for IAM	Added support for IAM. For more information, see Controlling access to your Amazon EC2 Auto Scaling resources .	2 December 2010