

---

# **Auto Scaling**

## **Developer Guide**

### **API Version 2011-01-01**



## Auto Scaling: Developer Guide

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, AWS CloudTrail, AWS CodeDeploy, Amazon Cognito, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Amazon Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC, and Amazon WorkDocs. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. 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 Auto Scaling?	1
Auto Scaling Components	2
Getting Started	2
Accessing Auto Scaling	2
Pricing for Auto Scaling	3
Related Services	3
Benefits of Auto Scaling	3
Example: Covering Variable Demand	4
Example: Web App Architecture	5
Example: Distributing Instances Across Availability Zones	6
Launch Configurations	7
Auto Scaling Groups	8
Scaling Plans	8
Auto Scaling Lifecycle	9
Auto Scaling Basic Lifecycle	9
Auto Scaling Instance States	11
Auto Scaling Limits	14
Setting Up	15
Sign Up for AWS	15
Prepare to Use Amazon EC2	15
Getting Started	16
Step 1: Create a Launch Configuration	16
Step 2: Create an Auto Scaling Group	17
Step 3: Verify Your Auto Scaling Group	18
Step 4: (Optional) Delete Your Auto Scaling Infrastructure	19
Planning Your Auto Scaling Group	21
Scaling Your Group	22
Multiple Scaling Policies	22
Understanding Cooldowns	23
Choosing a Termination Policy	26
Maintaining a Fixed Number of EC2 Instances	29
Manual Scaling	30
Dynamic Scaling	32
Controlling Access to Your Auto Scaling Resources	45
Auto Scaling Actions	46
Auto Scaling Resources	46
Auto Scaling Keys	46
Example IAM Policies for Auto Scaling	47
Launch Auto Scaling Instances with an IAM Role	48
Creating Launch Configurations	49
Create a Launch Configuration	49
Create a Launch Configuration Using an EC2 Instance	50
Creating Auto Scaling Groups	54
Create an Auto Scaling Group	55
Create an Auto Scaling Group from an EC2 Instance	56
Auto Scaling and Amazon VPC	58
Default VPC	59
IP Addressing in a VPC	59
Instance Placement Tenancy	59
Linking EC2-Classic Instances to a VPC	60
Launch Auto Scaling Instances in a VPC	62
Controlling Instances	62
Introducing Lifecycle Hooks	63
Lifecycle Hook Examples	67
Tagging Auto Scaling Groups and Instances	73

Tag Restrictions .....	73
Add or Modify Tags for Your Auto Scaling Group .....	74
Delete Tags .....	75
Launching Spot Instances in Your Auto Scaling Group .....	76
Launching Spot Instances Using the AWS Management Console .....	77
Launching Spot Instances Using the AWS CLI .....	80
Configuring Your Auto Scaling Groups .....	85
Load Balance Your Auto Scaling Group .....	86
Set Up a Scaled and Load-Balanced Application .....	87
Add an Elastic Load Balancing Health Check to your Auto Scaling Group .....	91
Expand Your Scaled and Load-Balanced Application to an Additional Availability Zone .....	92
Attach EC2 Instances to Your Auto Scaling Group .....	95
Attaching an Instance Using the AWS Management Console .....	95
Attaching an Instance Using the AWS CLI .....	96
Detach EC2 Instances From Your Auto Scaling Group .....	98
Detaching Instances Using the AWS Management Console .....	99
Detaching Instances Using the AWS CLI .....	99
Merging Auto Scaling Groups .....	101
Merge Zones Using the AWS CLI .....	101
Temporarily Removing Instances .....	103
Troubleshooting Instances .....	103
Updating or Modifying Instances .....	107
Suspend and Resume Processes .....	110
Auto Scaling Processes .....	110
Suspend and Resume Processes Using the AWS CLI .....	111
Shut Down Auto Scaling Processes Using the AWS CLI .....	112
Delete Your Auto Scaling Group .....	112
(Optional) Delete the Launch Configuration .....	112
(Optional) Delete the Load Balancer .....	112
(Optional) Delete CloudWatch Alarms .....	113
Monitoring Your Auto Scaling Instances .....	114
Amazon CloudWatch Alarms .....	114
Activating Detailed Instance Monitoring for Auto Scaling .....	115
Activating Basic Instance Monitoring for Auto Scaling .....	115
Auto Scaling Group Metrics .....	116
Auto Scaling Group Metrics Table .....	116
Dimensions for Auto Scaling Group Metrics .....	117
Health Checks .....	117
Set the Health State of An Instance .....	118
Getting Notifications When Your Auto Scaling Group Changes .....	119
Configure Amazon SNS .....	119
Configure Your Auto Scaling Group to Send Notifications .....	120
Test the Notification Configuration .....	121
Verify That You Received Notification of the Scaling Event .....	121
Delete the Notification Configuration .....	123
Logging Auto Scaling API Calls By Using AWS CloudTrail .....	123
Auto Scaling Information in CloudTrail .....	124
Understanding Auto Scaling Log File Entries .....	124
Troubleshooting .....	127
Retrieving an Error Message .....	127
Instance Launch Failure .....	129
The security group <name of the security group> does not exist. Launching EC2 instance failed. ....	130
The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. ....	130
The requested configuration is currently not supported. ....	130
AutoScalingGroup <Auto Scaling group name> not found. ....	131
The requested Availability Zone is no longer supported. Please retry your request .....	131

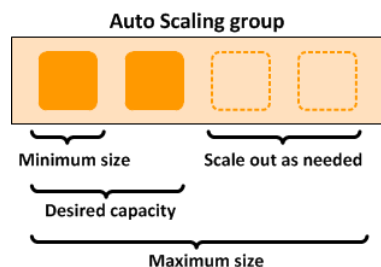
Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>).	131
You are not subscribed to this service. Please see <a href="http://aws.amazon.com">http://aws.amazon.com</a> .	131
Invalid device name upload. Launching EC2 instance failed.	131
Value (<name associated with the instance storage device>) for parameter virtualName is invalid.	132
EBS block device mappings not supported for instance-store AMIs.	132
Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.	132
AMI Issues	132
The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.	133
AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.	133
Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed.	133
Value (<ami ID>) for parameter virtualName is invalid.	133
The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching ec2 instance failed.	134
Load Balancer Issues	134
Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.	134
There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.	135
EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.	135
EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed.	135
The security token included in the request is invalid. Validating load balancer configuration failed.	135
Capacity Limits	136
We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>).	136
<number of instances> instance(s) are already running. Launching EC2 instance failed.	136
Command Line Interface	137
Resources	138
Document History	139
AWS Glossary	143

# What Is Auto Scaling?

---

Auto Scaling helps you ensure that you have the correct number of 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 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 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, Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then 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 1 instance, a desired capacity of 2 instances, and a maximum size of 4 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 Auto Scaling, see [Benefits of Auto Scaling \(p. 3\)](#).

# Auto Scaling Components

The following table describes the key components of Auto Scaling.

	<p><b>Groups</b></p> <p>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 <a href="#">Auto Scaling Groups (p. 8)</a>.</p>
	<p><b>Launch configurations</b></p> <p>Your group uses a <i>launch configuration</i> as a template for its EC2 instances. When you create a launch configuration, 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 <a href="#">Launch Configurations (p. 7)</a>.</p>
	<p><b>Scaling plans</b></p> <p>A <i>scaling plan</i> tells Auto Scaling when and how to scale. For example, you can base a scaling plan on the occurrence of specified conditions (dynamic scaling) or on a schedule. For more information, see <a href="#">Scaling Plans (p. 8)</a>.</p>

## Getting Started

If you're new to Auto Scaling, we recommend that you review [Auto Scaling Lifecycle \(p. 9\)](#) before you begin.

To begin, complete the [Getting Started with Auto Scaling \(p. 16\)](#) tutorial to create an Auto Scaling group and see how it responds when an instance in that group terminates. If you already have running EC2 instances, you can create an Auto Scaling group using an existing EC2 instance, and remove the instance from the group at any time. After you are familiar with how Auto Scaling works, read [Planning Your Auto Scaling Group \(p. 21\)](#) to learn how to make the most of Auto Scaling.

## Accessing Auto Scaling

AWS provides a web-based user interface, the AWS Management Console. If you've signed up for an AWS account, you can access Auto Scaling by signing into the AWS Management Console. To get started, select **EC2** from the console home page, and then select **Launch Configurations** from the navigation pane.

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

### **AWS Command Line Interface (CLI)**

Provides commands for a broad set of AWS products, and is supported on Windows, Mac, and Linux. To get started, see [AWS Command Line Interface User Guide](#). For more information about the commands for Auto Scaling, see [autoscaling](#) in the *AWS Command Line Interface Reference*.

### **AWS Tools for Windows PowerShell**

Provides commands for a broad set of AWS products for those who script in the PowerShell environment. To get started, see the [AWS Tools for Windows PowerShell User Guide](#). For more information about the cmdlets for Auto Scaling, see the [AWS Tools for Windows PowerShell Reference](#).

Auto Scaling provides a Query API. These requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named `Action`. For more information about the API actions for Amazon EC2, see [Actions](#) in the *Amazon EC2 API Reference*.

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

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

## Pricing for Auto Scaling

There are no additional fees with Auto Scaling, so it's easy to try it out and see how it can benefit your AWS architecture.

## Related Services

To automatically distribute incoming application traffic across multiple instances in your Auto Scaling group, use Elastic Load Balancing. For more information, see [Elastic Load Balancing Developer Guide](#).

To monitor basic statistics for your instances and Amazon EBS volumes, use Amazon CloudWatch. For more information, see the [Amazon CloudWatch Developer Guide](#).

To monitor the calls made to the Auto Scaling API for your account, including calls made by the AWS Management Console, command line tools, and other services, use AWS CloudTrail. For more information, see the [AWS CloudTrail User Guide](#).

## Benefits of Auto Scaling

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

- Better fault tolerance. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it.
- Better availability. You can configure Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
- Better cost management. 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 actually needed and terminating them when they aren't needed.



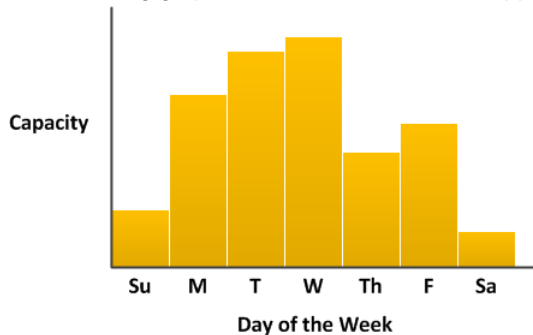
## Contents

- [Example: Covering Variable Demand \(p. 4\)](#)
- [Example: Web App Architecture \(p. 5\)](#)
- [Example: Distributing Instances Across Availability Zones \(p. 6\)](#)

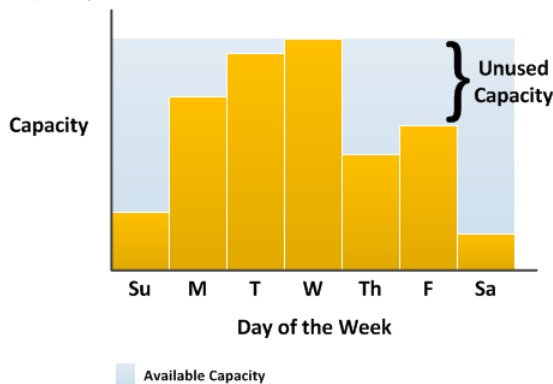
## Example: Covering Variable Demand

To demonstrate some of the benefits of 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 demands on the application increases significantly.

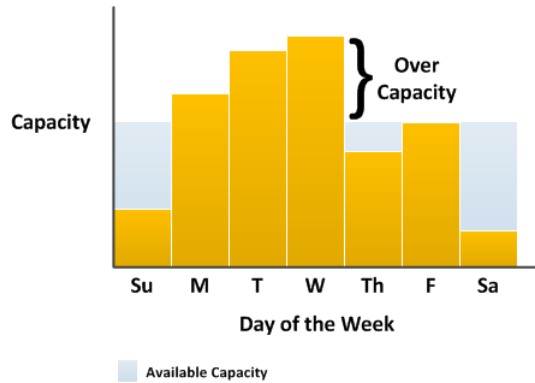
The following graph shows how much of the application's capacity is used over the course of a week.



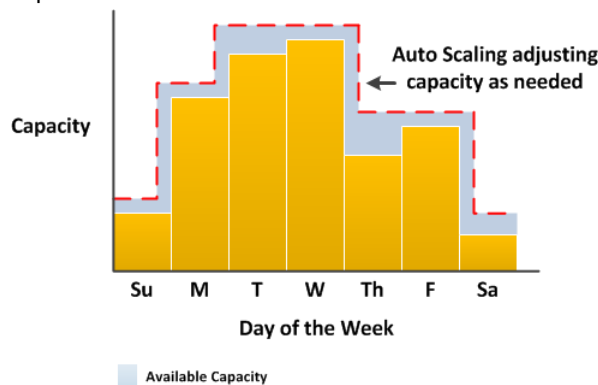
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity. 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 demands 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 demands on the application exceeds its capacity.

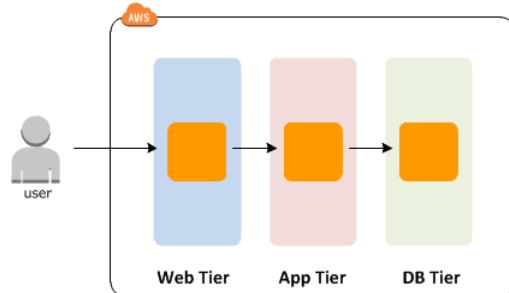


By adding Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. And because Auto Scaling uses EC2 instances, you only have to pay for the instances you use, when you use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

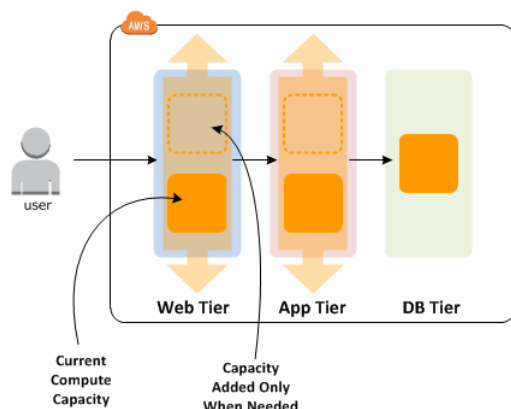


## Example: Web App Architecture

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.



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 can help you 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 [Load Balance Your Auto Scaling Group \(p. 86\)](#).

## Example: Distributing Instances Across Availability Zones

AWS resources, such as EC2 instances, are housed in highly-available data centers. To provide additional scalability and reliability, these data centers are in different physical locations. *Regions* are large and widely dispersed geographic locations. Each region contains multiple distinct locations, called *Availability Zones*, that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. For information about the regions for Auto Scaling, see [Regions and Endpoints: Auto Scaling](#) in the *Amazon Web Services General Reference*.

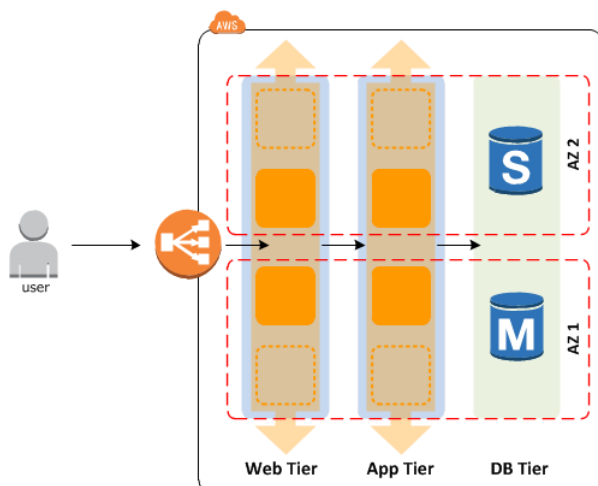
Auto Scaling enables you to take advantage of the safety and reliability of geographic redundancy by spanning Auto Scaling groups across multiple Availability Zones within a region. When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the designated Availability Zones.

An Auto Scaling group can contain EC2 instances in one or more Availability Zones within the same region. However, Auto Scaling groups cannot span multiple regions.

For Auto Scaling groups in a VPC, the EC2 instances are launched in subnets. You can create your VPC with one or more subnets in each Availability Zone. You select the subnets for your EC2 instances when you create or update the Auto Scaling group. For more information, see [Auto Scaling and Amazon Virtual Private Cloud \(p. 58\)](#).

## Instance Distribution

Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group. Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds. For each instance that Auto Scaling launches in a VPC, it selects a subnet from the Availability Zone at random.



## Rebalancing Activities

Certain operations and conditions can cause your Auto Scaling group to become unbalanced between Availability Zones. Auto Scaling compensates by creating a rebalancing activity under any of the following conditions:

- You issue a request to change the Availability Zones for your group.
- You explicitly call for termination of a specific instance that caused the group to become unbalanced.
- An Availability Zone that previously had insufficient capacity recovers and has additional capacity available.

When rebalancing, Auto Scaling launches new instances before terminating the old ones, so that rebalancing does not compromise the performance or availability of your application.

Because Auto Scaling attempts to launch new instances before terminating the old 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 by a 10 percent margin (or by a 1-instance margin, 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 typically a few minutes.

## Launch Configurations

A *launch configuration* is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.

When you create an Auto Scaling group, you must specify a launch configuration. 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. Therefore, if you want to change the launch configuration for your Auto Scaling group, you must create a new launch configuration and then update your Auto Scaling group with the new launch configuration. When you change the launch configuration for your Auto Scaling group, any new instances are launched using the new configuration parameters, but existing instances are not affected.

For information about creating a launch configuration, see [Creating Launch Configurations \(p. 49\)](#).

## Auto Scaling Groups

An *Auto Scaling group* contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping for the purposes of instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase the number of instances in that group to improve the performance of the application, or decrease the number of instances to reduce costs when demand is low. You can use the Auto Scaling group to scale the number of instances automatically based on criteria that you specify, or maintain a fixed number of instances even if a instance becomes unhealthy. This automatic scaling and maintaining the number of instances in an Auto Scaling group is the core value of the Auto Scaling service.

When you create a Auto Scaling group, you must specify a name, launch configuration, minimum number of instances, and maximum number of instances. You can optionally specify a desired capacity, which is the number of instances that the group must have at all times. If you don't specify a desired capacity, the default desired capacity is the minimum number of instances that you specified. For information about creating an Auto Scaling group, see [Creating Auto Scaling Groups \(p. 54\)](#).

An Auto Scaling group starts by launching enough EC2 instances to meet its desired capacity. The Auto Scaling group maintains this number of instances by performing periodic health checks on the instances in the group. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it.

You can use scaling policies to increase or decrease the number of running EC2 instances in your group automatically to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group and launches or terminates the instances as needed. If you manually scale or scale on a schedule, you must adjust the desired capacity of the group in order for the changes to take effect. For more information, see [Scaling Plans \(p. 8\)](#).

## Scaling Plans

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 minimum or specified number of running instances at all times. To maintain the current instance levels, Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one. For information about configuring your Auto Scaling group to maintain the current instance levels, see [Maintaining a Fixed Number of EC2 Instances in Your Auto Scaling Group \(p. 29\)](#).

### **Manual scaling**

Manual scaling is the most basic way to scale your resources. You only need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity. For more information, see [Manual Scaling \(p. 30\)](#).

### **Scale based on a schedule**

Sometimes you know exactly when you will need to increase or decrease the number of instances in your group, simply because that need arises on a predictable schedule. Scaling by schedule means that scaling

actions are performed automatically as a function of time and date. For more information, see [Scheduled Scaling](#) (p. 40).

### Scale based on demand

A more advanced way to scale your resources, scaling by policy, lets you define parameters that control the Auto Scaling process. For example, you can create a policy that calls for enlarging your fleet of EC2 instances whenever the average CPU utilization rate stays above ninety percent for fifteen minutes. This is useful when you can define how you want to scale in response to changing conditions, but you don't know when those conditions will change. You can set up Auto Scaling to respond for you.

Note that you should have two policies, one for scaling in (terminating instances) and one for scaling out (launching instances), for each event to monitor. For example, if you want to scale out when the network bandwidth reaches a certain level, create a policy specifying that Auto Scaling should start a certain number of instances to help with your traffic. But you may also want an accompanying policy to scale in by a certain number when the network bandwidth level goes back down. For more information, see [Dynamic Scaling](#) (p. 32).

## Auto Scaling Lifecycle

Like [Amazon EC2 instances launched manually](#), instances in an Auto Scaling group follow a specific path, or lifecycle. For Auto Scaling instances, this lifecycle starts when you [create a new Auto Scaling group](#) or when a [scale out event](#) occurs. At that point, a new instance launches and is put into service by the Auto Scaling group. The lifecycle ends when a corresponding scale in event occurs, at which point the Auto Scaling group detaches the instance and terminates it.

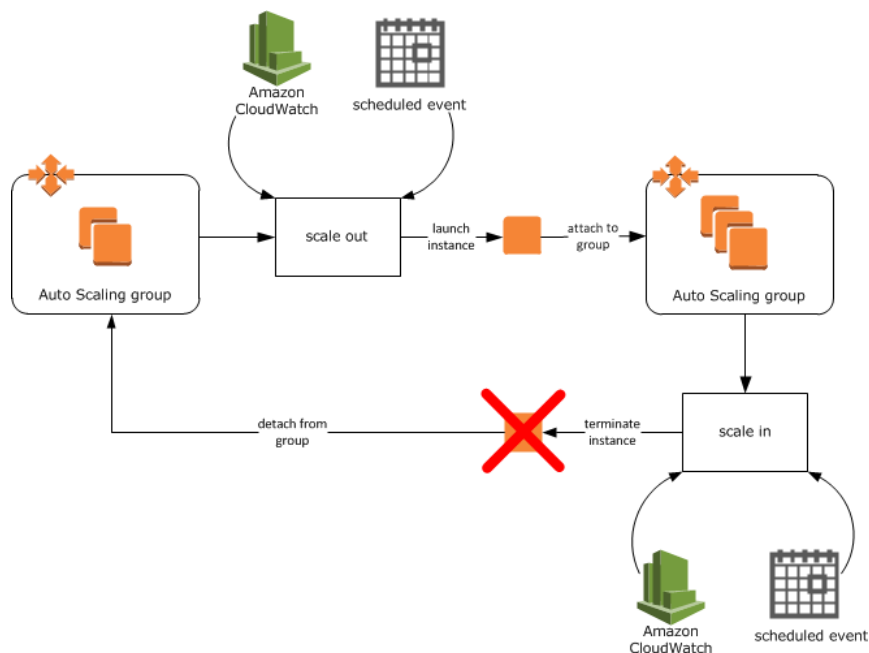
In the [Getting Started](#) topic, you can learn how the basic Auto Scaling group lifecycle is simple to implement and powerful to use. In the following sections, you'll also learn how you can fine-tune your implementation of Auto Scaling to best suit your applications' and your customers' expectations.

### Contents

- [Auto Scaling Basic Lifecycle](#) (p. 9)
- [Auto Scaling Instance States](#) (p. 11)

## Auto Scaling Basic Lifecycle

The following illustration shows the basic lifecycle of instances within an Auto Scaling group. The Auto Scaling group has a desired capacity of two instances, a CloudWatch alarm that can trigger scaling events, and policies that scale the group at specific dates and times.



Each part of the lifecycle has performance implications for your Auto Scaling group.

### Scale out

These events direct the Auto Scaling group to launch new instances and add them to the group. For example:

- You [manually](#) (p. 30) increase the number of instances, either by setting a new minimum number of instances or desired capacity for the group.
- You use a [Amazon CloudWatch alarm](#) (p. 32) to monitor your application and scale based on specified criteria.
- You use a [schedule-based policy](#) (p. 40) to increase or decrease the number of instances in the group at a specific time.
- An existing instance fails required health checks, or you [manually configure an instance](#) (p. 118) to have a have an `Unhealthy` status.

### Launch instances

After a scale out event occurs, the Auto Scaling group uses its assigned launch configuration to launch one or more EC2 instances. The number of instances launched depends on how you configured the scaling policies for your group. Instances that have launched but are not yet fully configured are typically in the [Pending](#) (p. 12) state. You have the option of adding a hook to your Auto Scaling group that puts instances in this state into a `Pending:Wait`. This state allows you to access these instances before they are put into service.

### Attach instances to the Auto Scaling group

After an instance is launched and fully configured, it is put into service and attached to the Auto Scaling group. The instance now counts against the minimum size, maximum size, and desired capacity (if set) for the Auto Scaling group. These instances are in the `InService` state.

### Scale in

These events direct the Auto Scaling group to terminate instances and detach them from the group. They can be triggered in the same way as a scale out event. It is important that you create a 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.

### Terminate instances

Finally, the instance is completely terminated.

### Detach instances from the Auto Scaling group

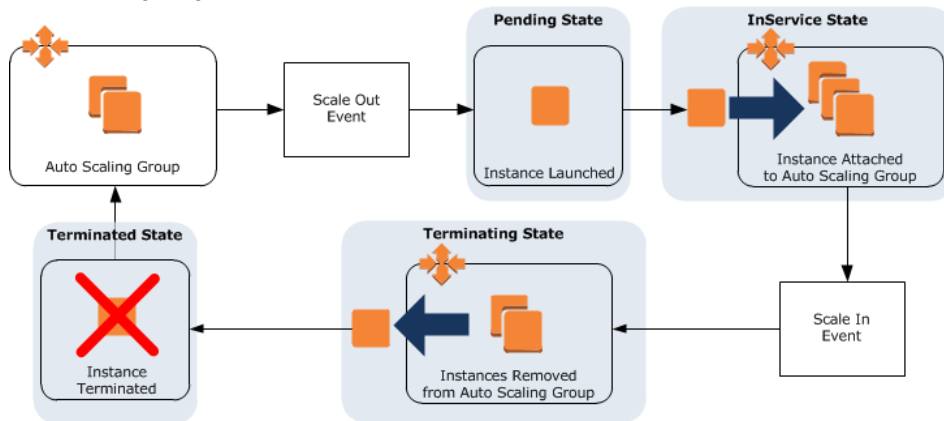
After a scale in event occurs, the Auto Scaling group detaches one or more instances. How the Auto Scaling group determines which instance to terminate depends on its [termination policy](#) (p. 26). Instances that are in the process of detaching from the Auto Scaling group and shutting down are in the [Terminating](#) (p. 13) state. You have the option of adding a hook to your Auto Scaling group instances in this state into a `Terminating:Wait` state. This state allows you to access these instances before they are terminated.

## Auto Scaling Instance States

Instances in an Auto Scaling group can be in one of four main states:

- [Pending](#) (p. 12)
- [InService](#) (p. 12)
- [Terminating](#) (p. 13)
- Terminated

The following diagram shows how an instance moves from one state to another.



You can take specific actions when an instance is in one of these states:

State	Action
Pending	<a href="#">Installing Software to Pending Instances</a> (p. 68) <a href="#">Filling a Cache of Servers</a> (p. 69)
InService	<a href="#">Updating or Modifying Instances in an Auto Scaling Group</a> (p. 107) <a href="#">Troubleshooting Instances in an Auto Scaling Group</a> (p. 103)

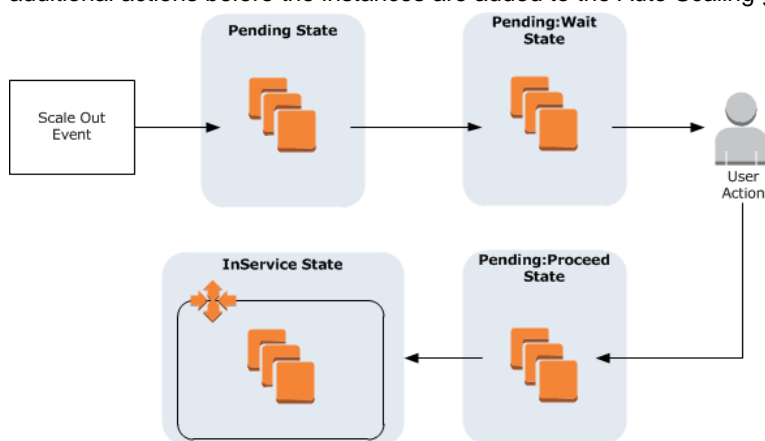


State	Action
Terminating	<a href="#">Analyzing an Instance Before Termination (p. 70)</a> <a href="#">Retrieving Logs from Terminating Instances (p. 72)</a>

## Auto Scaling Pending State

When an Auto Scaling group reaches a scale out threshold, it launches one or more instances (as determined by your scaling policy). These instances are configured based on the launch configuration for the Auto Scaling group. While an instance is launched and configured, it is in a `Pending` state.

Depending on how you want to manage your Auto Scaling group, the `Pending` state can be divided into two additional states: `Pending:Wait` and `Pending:Proceed`. You can use these states to perform additional actions before the instances are added to the Auto Scaling group.



Examples of these additional actions include:

- [Installing Software to Pending Instances \(p. 68\)](#)
- [Filling a Cache of Servers \(p. 69\)](#)

### Note

You are billed for instances as soon as they are launched. This means you will incur charges even if instances are in a `Pending:Wait` state but are not yet in service.

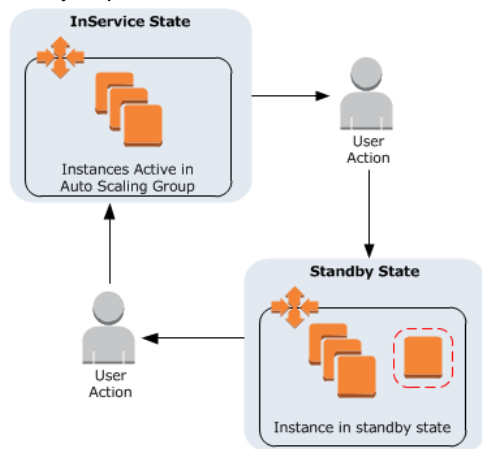
## Auto Scaling InService State

Instances that are functioning within your application as part of an Auto Scaling group are in the `InService` state. Instances remain in this state until:

- An Auto Scaling scale in event occurs, reducing the size of the Auto Scaling group
- You put the instance into a `Standby` state.
- You manually detach the instance from the Auto Scaling group
- The instance fails a required number of health checks or you manually set the status of the instance to `Unhealthy`.

In addition, any running instances that you attach to the Auto Scaling group are also in the `InService` state.

You have the option of putting any `InService` instance into a `Standby` state. Instances in this 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.



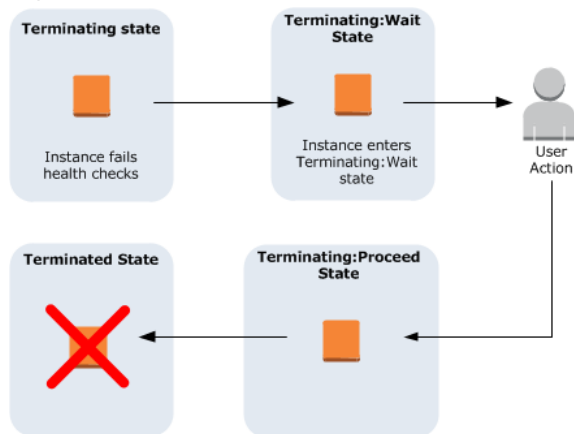
Examples of when you might put instances into the `Standby` state include:

- [To update or modify \(p. 107\)](#) the instance
- [To troubleshoot an instance \(p. 103\)](#) that isn't performing as expected

## Auto Scaling Terminating State

Instances that fail a required number of health checks are removed from an Auto Scaling group and terminated. The instances first enter the `Terminating` state, then `Terminated`.

Depending on how you want to manage your Auto Scaling group, the `Terminating` state can be divided into two additional states: `Terminating:Wait` and `Terminating:Proceed`. You can use these states to perform additional actions before the instances are terminated.



Examples of actions you can take while an instance is terminating include:

- [Analyzing an instance \(p. 70\)](#) to understand why it failed
- [Retrieving logs \(p. 72\)](#) stored on the instance.

### Important

You can use lifecycle hooks with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating due to a change in the Spot Price, which can happen at any time.

In addition, when a Spot Instance terminates, you must still complete the lifecycle action (such as with the **as-complete-lifecycle-action** command or **CompleteLifecycleAction** API call). For more information, see [Spot Instances](#).

## Auto Scaling Limits

Your AWS account comes with default limits on your AWS resources. To learn about the default limits for Auto Scaling, see [AWS Service Limits: Auto Scaling Limits](#).

If you reach the default limit for an AWS resource, you can request a limit increase. For more information, see [AWS Service Limits](#).

To view the current limit on your Auto Scaling resources, use the [describe-account-limits](#) (AWS CLI) command.

# Setting Up Auto Scaling

---

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

## Tasks

- [Sign Up for AWS](#) (p. 15)
- [Prepare to Use Amazon EC2](#) (p. 15)

## Sign Up for AWS

When you create an AWS account, we automatically sign up your account for all AWS services. You pay only for the services that you use. You can use Auto Scaling at no additional charge beyond what you are paying for your EC2 instances.

If you don't have an AWS account, sign up for AWS as follows.

### To sign up for an AWS account

1. Open <http://aws.amazon.com/>, and then click **Sign Up**.
2. Follow the on-screen instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

AWS sends you a confirmation e-mail after the sign-up process is complete.

## 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 Microsoft Windows Instances*, depending on which operating system you plan to use for your EC2 instances.

# Getting Started with Auto Scaling

---

Whenever you plan to use Auto Scaling, you must use certain building blocks to get started. This tutorial walks you through the process for setting up the basic infrastructure for Auto Scaling.

The following step-by-step instructions help you create a template that defines your EC2 instances, create an Auto Scaling group to maintain the healthy number of instances at all times, and optionally delete this basic Auto Scaling infrastructure. This tutorial assumes that you are familiar with launching EC2 instances and have already created a key pair and a security group.

## Tasks

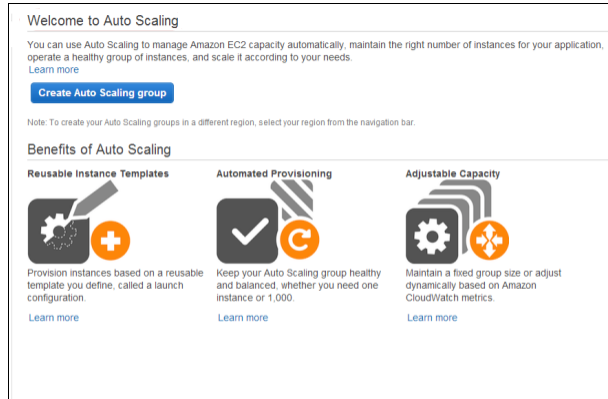
- [Step 1: Create a Launch Configuration \(p. 16\)](#)
- [Step 2: Create an Auto Scaling Group \(p. 17\)](#)
- [Step 3: Verify Your Auto Scaling Group \(p. 18\)](#)
- [Step 4: \(Optional\) Delete Your Auto Scaling Infrastructure \(p. 19\)](#)

## Step 1: Create a Launch Configuration

A launch configuration specifies the type of EC2 instance that Auto Scaling creates for you. You create the launch configuration by including information such as the Amazon Machine Image (AMI) ID to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings.

### To create a launch configuration

1. Open the Amazon EC2 console.
2. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**.
3. Select a region. The Auto Scaling resources that you create are tied to the region you specify and are not replicated across regions. For more information, see [Example: Distributing Instances Across Availability Zones \(p. 6\)](#).
4. On the **Welcome to Auto Scaling** page, click **Create Auto Scaling group**.



5. On the **Create Auto Scaling Group** page, click **Create launch configuration**.
6. On the **Choose AMI** page displays a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instance. Select the 64-bit Amazon Linux AMI.
7. On the **Choose Instance Type** page, select a hardware configuration for your instance. We recommend that you use the `t2.micro` instance that is selected by default. Click **Next: Configure details**.

**Note**

T2 instances must be launched into a subnet of a VPC. If you select a `t2.micro` instance but don't have a VPC, one is created for you. This VPC includes a public subnet in each Availability Zone in the region.
8. On the **Configure Details** page, do the following:
  - a. In the **Name** field, enter a name of your launch configuration (for example, `my-first-lc`).
  - b. Under **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to you instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.
  - c. Click **Skip to review**.
9. On the **Review** page, click **Edit security groups**, follow the instructions to choose an existing security group, and then click **Review**.
10. On the **Review** page, click **Create launch configuration**.
11. In the **Select an existing key pair or create a new key pair** dialog box, select one of the listed options. Note that you won't connect to your instance as part of this tutorial. Therefore, you can select **Proceed without a key pair** unless you intend to connect to your instance.
12. Click **Create launch configuration** to create your launch configuration.

## Step 2: Create an Auto Scaling Group

Auto Scaling groups are the core of the Auto Scaling service. An Auto Scaling group is a collection of EC2 instances. You create an Auto Scaling group by specifying the launch configuration you want to use for launching the instances and the number of instances your group must maintain at all times. You also specify the Availability Zone in which you want the instances to be launched.

### To create an Auto Scaling group

1. On the **Configure Auto Scaling group details** page, do the following:

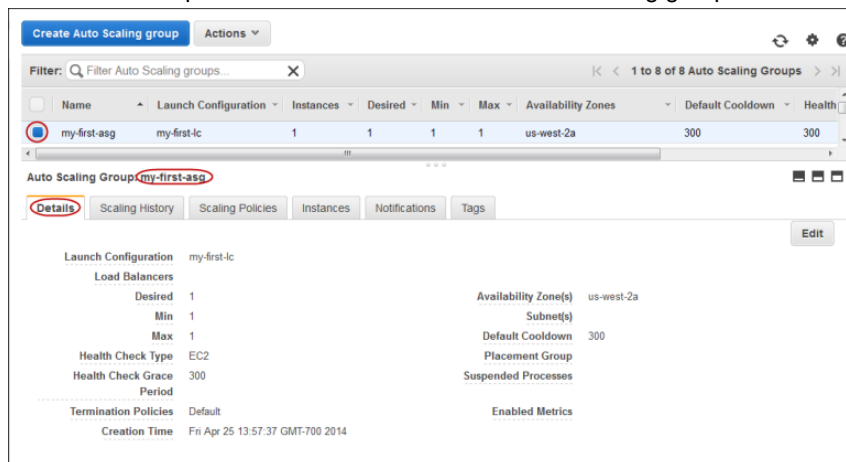
- a. In **Group name**, enter a name for your Auto Scaling group (for example, `my-first-asg`).
  - b. Leave **Group size** set to the default value of 1 instance for this tutorial.
  - c. If you are launching a `t2.micro` instance, you must select a VPC in **Network**. Otherwise, if your account supports EC2-Classic and you are launching a type of instance that doesn't require a VPC, you can select either `Launch into EC2-Classic` or a VPC.
  - d. If you selected a VPC in the previous step, select a subnet from **Subnet**. If you selected EC2-Classic in the previous step, select an Availability Zone from **Availability Zone(s)**.
  - e. Click **Next: Configure scaling policies**.
2. In the **Configure scaling policies** page, select **Keep this group at its initial size** for this tutorial and click **Review**.
  3. On the **Review** page, click **Create Auto Scaling group**.
  4. On the **Auto Scaling group creation status** page, click **Close**.

## Step 3: Verify Your Auto Scaling Group

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

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

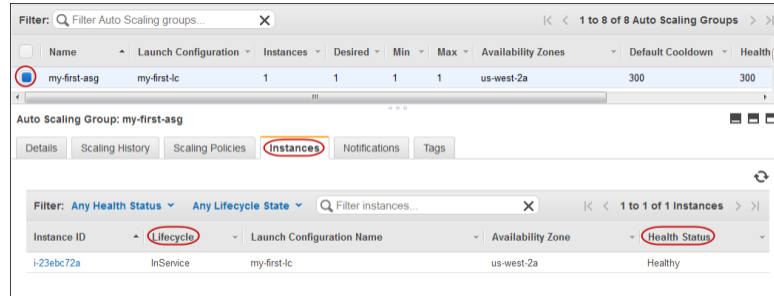
1. On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.
2. The **Details** tab provides information about the Auto Scaling group.



3. Select the **Scaling History** tab. The **Status** column contains the current status of your instance. When your instance is launching, the status column shows `In progress`. The status changes to `Successful` after the instance is launched. You can also click the refresh button to see the current status of your instance.
4. Select the **Instances** tab. The **Lifecycle** column contains the state of your newly launched instance. You can see that your Auto Scaling group has launched your EC2 instance, and it is in the `InService` lifecycle state. The **Health Status** column shows the result of the EC2 instance health check on your instance.

## Auto Scaling Developer Guide

### Step 4: (Optional) Delete Your Auto Scaling Infrastructure



5. (Optional) If you want, you can try the following experiment to learn more about Auto Scaling. The minimum size for your Auto Scaling group is 1 instance. Therefore, if you terminate the running instance, Auto Scaling must launch a new instance to replace it.
  - a. On the **Instances** tab, click the ID of the instance. This takes you to the **Instances** page and selects the instance.
  - b. Click **Actions**, select **Instance State**, and then click **Terminate**. When prompted for confirmation, click **Yes, Terminate**.
  - c. In the navigation pane, select **Auto Scaling Groups** and then select the **Scaling History** 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. When the scaling activity starts, you'll see an entry for the termination of the first instance and an entry for the launch of a new instance. The **Instances** tab shows the new instance only.
  - d. In the navigation pane, select **Instances**. This page shows both the terminated instance and the running instance.

Go to the next step if you would like to delete your Auto Scaling set up. Otherwise, you can use this Auto Scaling infrastructure as your base and try one or more of the following:

- [Maintaining a Fixed Number of EC2 Instances in Your Auto Scaling Group \(p. 29\)](#)
- [Manual Scaling \(p. 30\)](#)
- [Dynamic Scaling \(p. 32\)](#)
- [Getting Notifications When Your Auto Scaling Group Changes \(p. 119\)](#)

## Step 4: (Optional) Delete Your Auto Scaling Infrastructure

You can either delete your Auto Scaling set up or delete just your Auto Scaling group and keep your launch configuration to use at a later time.

### To delete your Auto Scaling group

1. Open the Amazon EC2 console.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group (for example, `my-first-asg`).
4. Click **Actions** and then click **Delete**. When prompted for confirmation, click **Yes, Delete**.

The **Name** column indicates that the Auto Scaling group is being deleted. The **Desired**, **Min**, and **Max** columns shows 0 instances for the Auto Scaling group.



Skip this procedure if you would like keep your launch configuration.

**To delete your launch configuration**

1. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**.
2. On the **Launch Configurations** page, select your launch configuration (for example, `my-first-lc`).
3. Click **Actions** and select **Delete launch configuration**. When prompted for confirmation, click **Yes, Delete**.

# Planning Your Auto Scaling Group

---

Auto Scaling, when correctly implemented, provides a number of advantages to your applications. An Auto Scaling group can help you make sure that your application always has the right amount of capacity to handle the current traffic demands. You can also use Auto Scaling to make your applications more highly available and fault tolerant. Most importantly, you can implement Auto Scaling at no additional cost—you only pay for the Amazon EC2 resources you use.

There are actions that you need to consider before you put your first Auto Scaling group into production. By planning ahead, you can help ensure that your Auto Scaling performs as expected and in a cost-effective manner.

Before you get started, take the time to review your application thoroughly as it runs in the AWS cloud. Take note of things like:

- How long it takes to launch and configure a server
- What metrics have the most relevance to your application's performance
- What existing resources (such as EC2 instances or AMIs) you might want to use as part of your Auto Scaling group
- How many Availability Zones you want the Auto Scaling group to span
- The role you want Auto Scaling to play in your application. Do you want Auto Scaling to use scaling to increase or decrease capacity? Or do you want to use it solely to ensure that a specific number of servers are always running? (Keep in mind that an Auto Scaling group can actually perform both functions simultaneously.)

The better you understand your application, the more effective your implementation of Auto Scaling becomes.

When you have enough information about your application, take a look at the section, [Scaling the Size of Your Auto Scaling Group \(p. 22\)](#). This section describes the different ways that Auto Scaling can help you adjust your application's capacity. In addition, this section describes features such as [Auto Scaling cooldowns \(p. 23\)](#) and [termination policies \(p. 26\)](#), which play important roles in controlling how Auto Scaling scales your application.

When you have a good idea of how you want to scale your architecture, the next section you should review is [Controlling Access to Your Auto Scaling Resources \(p. 45\)](#), which describes the role [AWS Identity and Access Management](#) plays in managing your EC2 instances in an Auto Scaling group.

## Contents

- [Scaling the Size of Your Auto Scaling Group \(p. 22\)](#)

- [Controlling Access to Your Auto Scaling Resources \(p. 45\)](#)
- [Creating Launch Configurations \(p. 49\)](#)
- [Creating Auto Scaling Groups \(p. 54\)](#)
- [Auto Scaling and Amazon Virtual Private Cloud \(p. 58\)](#)
- [Controlling How Instances Launch and Terminate \(p. 62\)](#)
- [Tagging Auto Scaling Groups and Instances \(p. 73\)](#)
- [Launching Spot Instances in Your Auto Scaling Group \(p. 76\)](#)

## Scaling 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 Auto Scaling to either launch or terminate EC2 instances.

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. You should keep the following considerations in mind:

- What role do you want Auto Scaling to play in your application's architecture? It's common to think about Auto Scaling as a way to increase and decrease capacity, but Auto Scaling is also useful for when you want to maintain a steady number of servers.
- What cost constraints are important to you? Because Auto Scaling uses EC2 instances, you only pay for the resources you use. Knowing your cost constraints can help you decide when to scale your applications, and by how much.
- What metrics are important to your application? CloudWatch supports a number of different metrics that you can use with your Auto Scaling group. We recommend reviewing them to see which of these metrics are the most relevant to your application.

To learn more about scaling implementations, see the following:

### [Cooldowns \(p. 23\)](#)

Periods of time during which Auto Scaling ignores any additional scaling actions.

### [Termination policies \(p. 26\)](#)

Criteria that determine which instances Auto Scaling should terminate first.

### [Maintaining a Fixed Number of EC2 Instances in Your Auto Scaling Group \(p. 29\)](#)

Maintains the minimum or specified number of instances in your Auto Scaling group at all times.

### [Manual Scaling \(p. 30\)](#)

Change the number of running instances in your Auto Scaling group manually at any time.

### [Dynamic Scaling \(p. 32\)](#)

Scale dynamically in response to changes in the demand for your application. You must specify when and how to scale.

### [Scheduled Scaling \(p. 40\)](#)

Scale your application on a predefined schedule (one-time only or on a recurring schedule).

## Multiple Scaling Policies

An Auto Scaling group can have more than one scaling policy attached to it any given time. In fact, we recommend that each Auto Scaling group has at least two policies: one to scale your architecture out and another to scale your architecture in. You can also combine scaling policies to maximize the performance of an Auto Scaling group.

To illustrate how multiple policies work together, consider an application that uses an Auto Scaling group and an Amazon SQS queue to send requests to the EC2 instances in that group. To help ensure the application performs at optimum levels, there are two policies that control when the Auto Scaling group should scale out. One policy uses the Amazon CloudWatch metric, `CPUUtilization`, to detect when an instance is at 90% of capacity. The other uses the `NumberOfMessagesVisible` to detect when the SQS queue is becoming overwhelmed with messages.

**Note**

In a production environment, both of these policies would have complementary policies that control when Auto Scaling should scale in the number of EC2 instances.

When you have more than one policy attached to an Auto Scaling group, there's a chance that both policies could instruct Auto Scaling to scale out (or in) at the same time. In our previous example, it's possible that both an EC2 instance could trigger the CloudWatch alarm for the `CPUUtilization` metric, and the SQS queue trigger the alarm for the `NumberOfMessagesVisible` metric.

When these situations occur, Auto Scaling chooses the policy that has the greatest impact on the Auto Scaling group. For example, suppose that the policy for CPU utilization instructs Auto Scaling to launch 1 instance, while the policy for the SQS queue prompts Auto Scaling to launch 2 instances. If the scale out criteria for both policies are met at the same time, Auto Scaling gives precedence to the SQS queue policy, because it has the greatest impact on the Auto Scaling group. This results in Auto Scaling launching two instances into the group. This precedence applies even when the policies use different criteria for scaling out. For instance, if one policy instructs Auto Scaling to launch 3 instances, and another instructs Auto Scaling to increase capacity by 25 percent, Auto Scaling give precedence to whichever policy has the greatest impact on the group at that time.

## Understanding Auto Scaling Cooldowns

As described in [What Is Auto Scaling? \(p. 1\)](#), you can use Auto Scaling groups to scale—increase and decrease—the resources available to your application. You have a variety of different scaling methods available to you, such as [manual scaling \(p. 30\)](#) or [dynamic scaling \(p. 32\)](#). Regardless of how you decide to scale your resources, you need to consider the Auto Scaling cooldown period and how you want it to affect your Auto Scaling group.

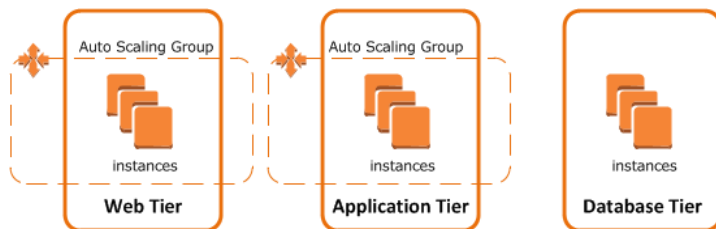
The Auto Scaling cooldown period is a configurable setting that determines when Auto Scaling should suspend any scaling activities related to a specific Auto Scaling group. This cooldown period is important, because it helps to ensure you don't launch or terminate more resources than you need.

**Contents**

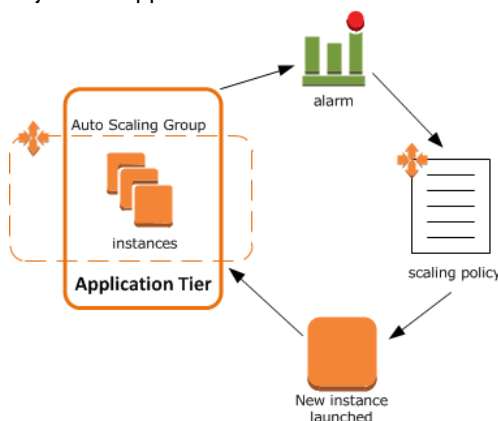
- [Example: Auto Scaling Cooldowns \(p. 23\)](#)
- [Default Cooldowns \(p. 24\)](#)
- [Scaling-Specific Cooldowns \(p. 25\)](#)
- [Cooldowns and Multiple Instances \(p. 26\)](#)
- [Cooldowns and Lifecycle Hooks \(p. 26\)](#)
- [Cooldowns and Spot Instances \(p. 26\)](#)

## Example: Auto Scaling Cooldowns

Consider the following scenario: you have a web application running in AWS. This web application consists three basic tiers: web, application, and database. To make sure that the application always has the resources it needs to meet traffic demands, you create two Auto Scaling groups: one for your web tier and one for your application tier.



To help make sure the Auto Scaling group for the application tier has the appropriate amount of resources available, you [create an CloudWatch alarm](#) to occur whenever the [CPUUtilization metric](#) for the EC2 instances exceeds 90%. When the alarm occurs, Auto Scaling launches and configures another instance to join the application tier.



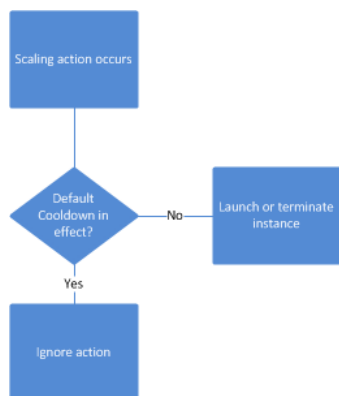
These instances use a configuration script to install and configure software before the instance is put into service. As a result, it takes around two or three minutes from the time the instance launches to when it is in service. (The actual time, of course, depends on several factors, such as whether you are using an AMI, the size of the instance, and so on.)

Now a spike in traffic occurs, causing the CloudWatch alarm to fire. When it does, Auto Scaling launches an instance to help handle the increase in demand. However, there's a problem: the instance takes a couple of minutes to launch. During that time, the CloudWatch alarm could continue to fire, resulting in Auto Scaling launch another instance each time the alarm goes off.

This is where the cooldown period comes into effect. With a cooldown period in place, Auto Scaling launches an instance and then suspends any scaling activities until a specific amount of time elapses. (The default amount of time is 300 seconds.) This way, the newly-launched instance has time to start handling application traffic. After the cooldown period expires, scaling actions resume for the Auto Scaling group. If the CloudWatch alarm is still occurring, Auto Scaling launches another instance, and the cooldown period takes effect again. If, however, the additional instance was enough to bring the CPU utilization back down, then the group remains at its current size.

## Default Cooldowns

As illustrated in the previous example, an Auto Scaling cooldown period helps to ensure you don't launch or terminate more resources than your application needs. Auto Scaling supports two types of cooldown periods: a default cooldown period and a [scaling-specific \(p. 25\)](#) cooldown period.



The default cooldown period is applied when you create your Auto Scaling group. Its default value is 300 seconds. This cooldown period applies to any scaling activity that occurs within the Auto Scaling group. This means it not only applies to [dynamic scaling \(p. 32\)](#) actions, as described the preceding section, but [manual scaling \(p. 30\)](#) actions as well.

You can configure the default cooldown period when you create the Auto Scaling group, using any of the following:

- AWS Management Console
- AWS CLI ([create-auto-scaling-group](#))
- [CreateAutoScalingGroup](#) API

You can change the default cooldown period whenever you need to, using any of the following:

- AWS Management Console
- AWS CLI ([update-auto-scaling-group](#))
- [UpdateAutoScalingGroup](#) API

## Scaling-Specific Cooldowns

In addition to the default cooldown period, you can create cooldowns that apply to a specific scaling policy. Any cooldown period that you configure with a scaling policy automatically overrides the [default cooldown \(p. 24\)](#) period.

Having a scaling-specific cooldown period can be very helpful in a number of situations. One common implementation is with a scale in policy—a policy that terminates instances based on a specific criteria or metric.

### Note

It is always recommended that every Auto Scaling group that you create have at least two policies: one that scales out the number of instances in the group, and one that scales in the number of instances.

Consider the example described in [the section called “Example: Auto Scaling Cooldowns” \(p. 23\)](#). Let's say that, in addition to a policy that scales out, or increases, the number of instances in the Auto Scaling group, there is also a policy that scales in when the CPU Utilization metric falls below a 50%. Because this policy terminates instances, less time is needed to determine whether to terminate additional instances in the Auto Scaling group. The default cooldown period of 300 seconds is too long—costs can be reduced by applying a scaling-specific cooldown period of 180 seconds.

You can create a scaling-specific cooldown period using one of the following:

- AWS Management Console
- AWS CLI ([put-scaling-policy](#))
- [PutScalingPolicy](#) API

## Cooldowns and Multiple Instances

The preceding sections have provided examples that show how cooldown periods affect Auto Scaling groups when a single instance launches or terminates. However, it is not uncommon for Auto Scaling groups to launch more than one instance at a time. For example, you might choose to have Auto Scaling launch three instances when a specific metric threshold is met.

In these situations, the cooldown period (either the default cooldown or the scaling-specific cooldown) take effect starting when the last instance launches.

## Cooldowns and Lifecycle Hooks

Auto Scaling supports adding lifecycle hooks to Auto Scaling groups. These hooks allow you to [control how instances launch and terminate](#) (p. 62) within an Auto Scaling group, allowing you to perform actions on the instance before the instance is put into service or before it terminates.

These hooks can affect the impact of any cooldown periods configured for the Auto Scaling group or a scaling policy. If the instance remains in a wait state, any additional scaling actions for the Auto Scaling group are suspended. The cooldown period for the Auto Scaling group does not begin until after the instance moves out of the wait state.

## Cooldowns and Spot Instances

You can create Auto Scaling groups to use [Spot Instances](#) (p. 76) instead of On-demand or Reserved Instances. In these situations, the cooldown periods for the Auto Scaling group take effect when the bid for any Spot Instance is successful.

# Choosing a Termination Policy for Your Auto Scaling Group

With each Auto Scaling group, you control when Auto Scaling adds instances (referred to as *scaling out*) or remove instances (referred to as *scaling in*) from your network architecture. You can scale the size of your group manually by attaching and detaching instances, or you can automate the process through the use of a scaling policy.

When you have Auto Scaling automatically scale in, you must decide which instances Auto Scaling should terminate first. You can configure this through the use of a termination policy.

### Contents

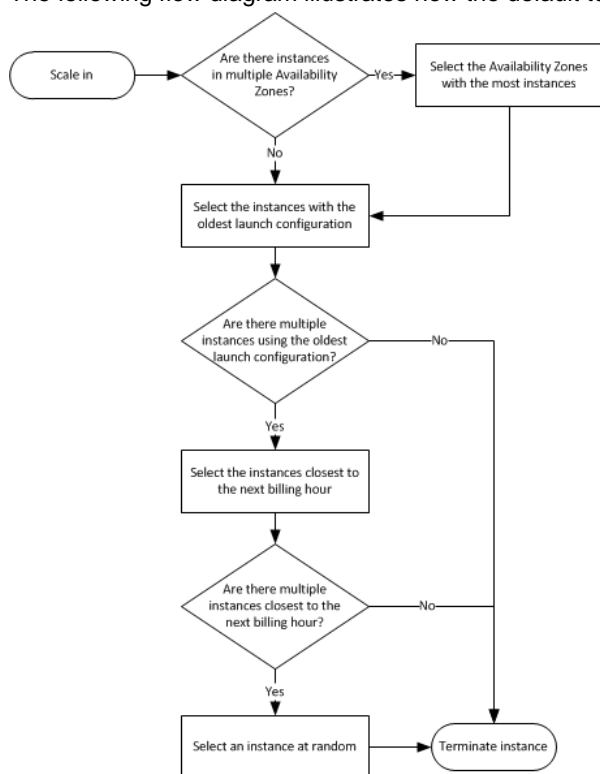
- [Default Termination Policy](#) (p. 26)
- [Customizing the Termination Policy](#) (p. 28)

## Default Termination Policy

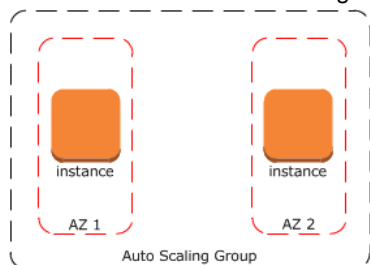
The default termination policy is designed to help ensure that your network architecture spans Availability Zones evenly. When using the default termination policy, Auto Scaling selects an instance to terminate as follows:

1. Auto Scaling determines whether there are instances in multiple Availability Zones. If so, it selects the Availability Zone with the most instances. If there is more than one Availability Zone with this number of instances, Auto Scaling selects the Availability Zone with the instances that use the oldest launch configuration.
2. Auto Scaling determines which instances in the selected Availability Zone use the oldest launch configuration. If there is one such instance, it terminates it.
3. If there are multiple instances that use the oldest launch configuration, Auto Scaling determines which instances are closest to the next billing hour. (This helps you maximize the use of your EC2 instances while minimizing the number of hours you are billed for Amazon EC2 usage.) If there is one such instance, Auto Scaling terminates it.
4. If there is more than one instance closest to the next billing hour, Auto Scaling selects one of these instances at random.

The following flow diagram illustrates how the default termination policy works.

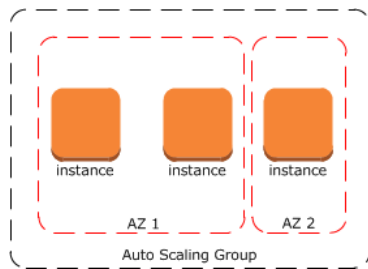


Consider an Auto Scaling group that has two Availability Zones, a desired capacity of two instances, and scaling policies that increase and decrease the number of instances by 1 when certain thresholds are met. The two instances in this group are distributed as follows.



When the threshold for the scale out policy is met, the policy takes effect and Auto Scaling launches a new instance. The Auto Scaling group now has three instances, distributed as follows.





When the threshold for the scale in policy is met, the policy takes effect and Auto Scaling terminates one of the instances. If the group does not have a specific termination policy assigned to it, Auto Scaling uses the default termination policy. Auto Scaling selects the Availability Zone with two instances, and terminates the instance launched from the oldest launch configuration. If the instances were launched from the same launch configuration, then Auto Scaling selects the instance that is closest to the next billing hour and terminates it.

## Customizing the Termination Policy

The default termination policy assigned to an Auto Scaling group is typically sufficient for most situations. However, you have the option of replacing the default policy with a customized one.

When you customize the termination policy, Auto Scaling first assesses the Availability Zones for any imbalance. If an Availability Zone has more instances than the other Availability Zones that are used by the group, then Auto Scaling applies your specified termination policy on the instances from the imbalanced Availability Zone. If the Availability Zones used by the group are balanced, then Auto Scaling applies the termination policy that you specified.

Auto Scaling currently supports the following custom termination policies:

- **OldestInstance.** Auto Scaling terminates 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, and want to eventually replace instances with older instances with newer ones.
- **NewestInstance.** Auto Scaling terminates 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.
- **OldestLaunchConfiguration.** Auto Scaling terminates 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.
- **ClosestToNextInstanceHour.** Auto Scaling terminates instances that are closest to the next billing hour. This policy helps you maximize the use of your instances and manage costs.
- **Default.** Auto Scaling uses its default termination policy. This policy is useful when you have more than one scaling policy associated with the group.

### To customize a termination policy using the console

1. Create the Auto Scaling group. For more information, see [Creating Auto Scaling Groups \(p. 54\)](#).
2. In the navigation pane, choose **Auto Scaling Groups**.
3. Select the group to update.
4. For **Actions**, choose **Edit**.
5. On the **Details** tab, locate **Termination Policies**. Choose one or more termination policies.
6. Choose **Save**.

### To customize a termination policy using the AWS CLI

Use one of the following commands:

- [create-auto-scaling-group](#)
- [update-auto-scaling-group](#)

You can use these policies individually, or combine them into a list of policies that Auto Scaling uses when terminating instances. For example, use the following command to update an Auto Scaling group to use the `OldestLaunchConfiguration` policy first, and then to 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 sure it's the last one in the list of termination policies. For example, `--termination-policies "OldestLaunchConfiguration,Default"`.

## Maintaining a Fixed Number of EC2 Instances in Your Auto Scaling Group

After you have created your launch configuration and Auto Scaling group, the Auto Scaling group starts by launching the minimum number of EC2 instances (or the desired number, if specified). If there are no other scaling conditions attached to the Auto Scaling group, the Auto Scaling group maintains this number of running instances at all times.

To maintain the same number of instances, Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When it finds that an instance is unhealthy, it terminates that instance and launches a new one.

All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless Auto Scaling receives notification that they are unhealthy. This notification can come from one or more of the following sources: Amazon EC2, Elastic Load Balancing, or your customized health check.

### Determining Instance Health

By default, the Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks. If the instance status is any state other than `running` or if the system status is `impaired`, Auto Scaling considers the instance to be unhealthy and launches a replacement. For more information about EC2 instance status checks, see [Monitoring the Status of Your Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you have associated your Auto Scaling group with a load balancer and have chosen to use the Elastic Load Balancing health check, Auto Scaling determines the health status of the instances by checking the results of both EC2 instance status and Elastic Load Balancing instance health. Auto Scaling marks an instance unhealthy if the instance is in a state other than `running`, the system status is `impaired`, or Elastic Load Balancing reports the instance state as `OutOfService`. To learn more about Elastic Load Balancing health checks, see [Elastic Load Balancing Health Check](#) in the *Elastic Load Balancing Developer Guide*.

You can customize the health check conducted by your Auto Scaling group by specifying additional checks, or if you have your own health check system, you can send the instance's health information directly from your system to Auto Scaling.

## Replacing Unhealthy Instances

After an instance has been marked unhealthy as a result of an Amazon EC2 or Elastic Load Balancing health check, it is almost immediately scheduled for replacement. It never automatically recovers its health. You can intervene manually by calling the [SetInstanceHealth](#) action (or the `as-set-instance-health` command) to set the instance's health status back to healthy, but you will get an error if the instance is already terminating. Because the interval between marking an instance unhealthy and its actual termination is so small, attempting to set an instance's health status back to healthy with the `SetInstanceHealth` action (or, `as-set-instance-health` command) is probably useful only for a suspended group. For more information, see [Suspend and Resume Auto Scaling Processes](#) (p. 110).

Auto Scaling creates a new scaling activity for terminating the unhealthy instance and then terminates it. Subsequently, another scaling activity launches a new instance to replace the terminated instance.

When your instance is terminated, any associated Elastic IP addresses are disassociated and are not automatically associated with the new instance. You must associate these Elastic IP addresses with the new instance manually. Similarly, when your instance is terminated, its attached EBS volumes are detached. You must attach these EBS volumes to the new instance manually.

## Manual Scaling

At any time, you can manually change the size of an existing Auto Scaling group. Auto Scaling manages the process of launching or terminating instances to maintain the updated group size.

### Prerequisites

The following examples assume 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.

### Contents

- [Scaling Manually Using the Console](#) (p. 30)
- [Scaling Manually Using the AWS CLI](#) (p. 31)

## Scaling Manually Using the Console

### To change the size of your Auto Scaling group manually

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the **Auto Scaling Groups** page, select your Auto Scaling group from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab and then click **Edit**.
5. In **Desired**, 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. Therefore, you must update **Max** if your new value for **Desired** is greater than **Max**.

When you are finished, click **Save**.

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. In the description pane of your Auto Scaling group, click the **Scaling History** tab.
2. The **Status** column lets you know that the current status of your instance. You can click the refresh button until you see the status of your new instance change to **Successful**, indicating that your Auto Scaling group has successfully launched a new instance.
3. Click the **Instances** tab.
4. On the **Instances** view pane, you can view the current **Lifecycle** state of your newly launched instances. It takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to **InService**. You can see that your Auto Scaling group has launched 1 new instance, and it is in the **InService** state.

## Scaling Manually Using the AWS CLI

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 --honor-cooldown
```

By default, the command overrides any cooldown period specified for the Auto Scaling group. You can choose to reject the default behavior and honor the cooldown period by specifying the `--honor-cooldown` option with the command. For more information, see [Understanding Auto Scaling Cooldowns \(p. 23\)](#).

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
```

Auto Scaling responds with details about the group and instances launched. The response should be similar to the following example:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 300,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-asg",
      "DefaultCooldown": 300,
      "MinSize": 1,
      "Instances": [
        {
          "InstanceId": "i-33388a3f",
          "AvailabilityZone": "us-west-2a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "my-lc"
        }
      ]
    }
  ],
}
```

```
        "MaxSize": 5,  
        "VPCZoneIdentifier": "subnet-e4f33493",  
        "TerminationPolicies": [  
            "Default"  
        ],  
        "LaunchConfigurationName": "my-lc",  
        "CreatedTime": "2014-12-12T23:30:42.611Z",  
        "AvailabilityZones": [  
            "us-west-2a"  
        ],  
        "HealthCheckType": "EC2"  
    }  
]  
}
```

Notice that `DesiredCapacity` shows the new value. Your Auto Scaling group has launched an additional instance.

## Dynamic Scaling

When you use Auto Scaling to scale on demand, you must define how you want to scale in response to changing conditions. For example, you have a web application that currently runs on two instances. You want to launch two additional instances when the load on the running instances reaches 70 percent, and then you want to terminate the additional instances when the load goes down to 40 percent. You can configure your Auto Scaling group to scale up and then scale down automatically based on specifying these conditions.

An Auto Scaling group uses a combination of policies and alarms to determine when the specified conditions for launching and terminating instances are met. An *alarm* is an object that watches over a single metric (for example, the average CPU utilization of your EC2 instances in an Auto Scaling group) over a time period that you specify. When the value of the metric breaches the thresholds that you define, over a number of time periods that you specify, the alarm performs one or more actions. An action can be sending messages to Auto Scaling. A *policy* is a set of instructions for Auto Scaling that tells the service how to respond to alarm messages.

Along with creating a launch configuration and Auto Scaling group, you need to create the alarms and the scaling policies and associate them with your Auto Scaling group. When the alarm sends the message, Auto Scaling executes the associated policy on your Auto Scaling group to scale the group in (terminate instances) or scale the group out (launch instances).

Auto Scaling integrates with CloudWatch for identifying metrics and defining alarms. For more information, see [Creating CloudWatch Alarms](#) in the *Amazon CloudWatch Developer Guide*.

### Contents

- [Scaling Policies](#) (p. 32)
- [Architectural Overview of Dynamic Scaling](#) (p. 33)
- [Scaling Based on Metrics](#) (p. 34)
- [Scheduled Scaling](#) (p. 40)
- [Scaling Based on Amazon SQS](#) (p. 42)

## Scaling Policies

When a scaling policy is executed, it changes the size of your Auto Scaling group by the amount specified in the policy. You can express the change to the current size as an absolute number, an increment, or

as a percentage of the current group size. When the policy is executed, Auto Scaling uses both the current group capacity and the change specified in the policy to compute a new size for your Auto Scaling group. Auto Scaling then updates the current size, and this consequently affects the size of your group.

A positive adjustment value increases the current capacity and a negative adjustment value decreases the current capacity. Auto Scaling does not scale above the maximum size or below the minimum size of the Auto Scaling group.

We recommend that you create two policies for each scaling change that you want to perform: one policy for scaling out and another policy for scaling in.

To create a scaling policy, you need to specify a name for the policy, the name of the Auto Scaling group to associate the policy with, the number of instances by which to scale, and the adjustment type. Auto Scaling supports the following adjustment types:

- **ChangeInCapacity:** Increases or decreases the existing capacity. For example, the current capacity of your Auto Scaling group is set to three instances, and you then create a scaling policy on your Auto Scaling group, specify the type as `ChangeInCapacity`, and the adjustment as five. When the policy is executed, Auto Scaling adds five more instances to your Auto Scaling group. You then have eight running instances in your Auto Scaling group: current capacity (3) plus `ChangeInCapacity` (5) equals 8.
- **ExactCapacity:** Changes the current capacity to the specified value. For example, if the current capacity is 5 instances and you create a scaling policy on your Auto Scaling group, specify the type as `ExactCapacity` and the adjustment as 3. When the policy is executed, your Auto Scaling group has three running instances.

You'll get an error if you specify a negative adjustment value for the `ExactCapacity` adjustment type.

- **PercentChangeInCapacity:** Increases or decreases the capacity by a percentage. For example, if the current capacity is 10 instances and you create a scaling policy on your Auto Scaling group, specify the type as `PercentChangeInCapacity`, and the adjustment as 10. When the policy is executed, your Auto Scaling group has eleven running instances because 10 percent of 10 instances is 1 instance, and 10 instances plus 1 instance is 11 instances.

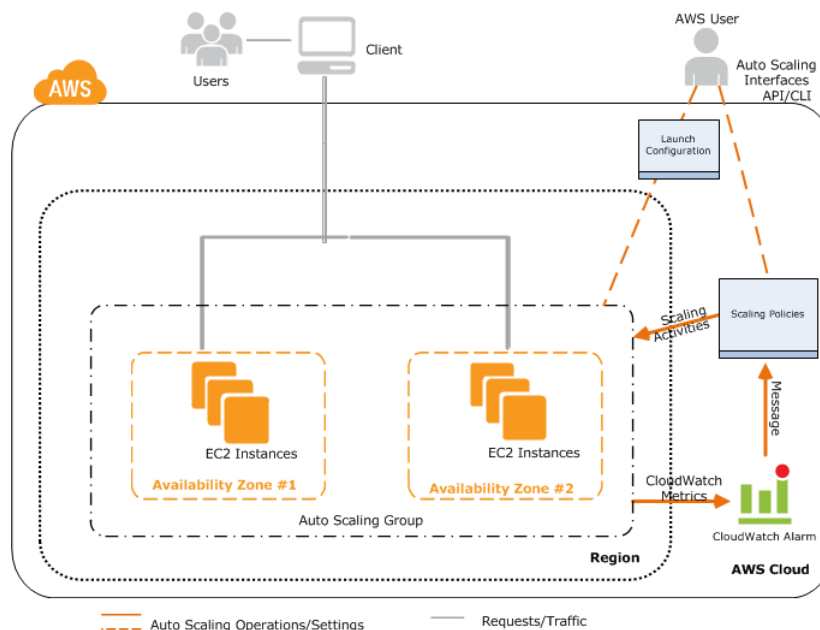
Auto Scaling handles non-integer numbers returned by `PercentChangeInCapacity` as follows:

- If the value is greater than 1, Auto Scaling rounds it to the lower value. For example, a return value of 12.7 is rounded to 12.
- If the value is between 0 and 1, Auto Scaling rounds it to 1. For example, a return value of .67 is rounded to 1.
- If the value between 0 and -1, Auto Scaling rounds it to -1. For example, a return value of -.58 is rounded to -1.
- If the value is less than -1, Auto Scaling rounds it to the higher value. For example, a return value of -6.67 is rounded to -6.

## Architectural Overview of Dynamic Scaling

The following diagram shows how the various components of Auto Scaling work together when you scale dynamically based on demand. The AWS user has done the following:

- Created a launch configuration by providing all the information required to launch EC2 instances.
- Created an Auto Scaling group by defining maximum, minimum, and (optionally), the desired capacity for the EC2 instances.
- Created an CloudWatch alarm and defined which metrics to monitor.
- Created two scaling policies, one for scaling out and another for scaling in, and associated the policies with the alarm.
- Associated the scaling policies with the Auto Scaling group.



The following events begin when a client sends a request to the AWS user's application, and with the launch of EC2 instances in the Auto Scaling group:

1. The application is ready to communicate with users after the Auto Scaling group has launched all Amazon EC2 application instances for the application.
2. While requests are being sent by users and received by the application instances, CloudWatch monitors the specified metrics of all the instances in the Auto Scaling group.
3. As the demand for the application either grows or shrinks, the specified metrics change.
4. The change in metrics invokes the CloudWatch alarm to perform an action. The action is a message sent to either the scaling-in policy or the scaling-out policy, depending on the metrics that were breached.
5. The Auto Scaling policy that receives the message then invokes the scaling activity within the Auto Scaling group.
6. This Auto Scaling process continues until the policies are deleted or the Auto Scaling group is terminated.

## Scaling Based on Metrics

When you create your policy, you can create CloudWatch alarms to watch specified scale-in and scale-out metrics. Then you associate the alarms with the scaling policies that you have created. These alarms send messages to Auto Scaling when the specified metrics breach the thresholds that you specified in your policies.

When you create your CloudWatch alarm, you can add an Amazon SNS topic to send an email notification when the alarm changes state. For more information, see the [Amazon Simple Notification Service Getting Started Guide](#).

Scaling policies also enable you to specify a custom cooldown period. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. Because scaling activities are suspended when an Auto Scaling group is in cooldown mode, an adequate cooldown period helps to prevent initiating a scaling activity based on stale metrics. By default, Auto Scaling uses a default period associated with your Auto Scaling group. When specified, the policy cooldown period takes priority over the default cooldown period specified in the Auto Scaling group. If the policy

does not specify a cooldown period, the group's default cooldown period is used. For more information, see [Understanding Auto Scaling Cooldowns](#) (p. 23).

The following examples use `CPUUtilization` metrics.

#### Contents

- [Scaling with Metrics Using the AWS Management Console](#) (p. 35)
- [Scaling with Metrics Using the AWS CLI](#) (p. 38)

## Scaling with Metrics Using the AWS Management Console

Use the AWS Management Console to configure scaling policies for your Auto Scaling group.

#### Tasks

- [Create an Auto Scaling Group](#) (p. 35)
- [Create Scaling Policies](#) (p. 36)
- [Verify Your Scaling Policies and CloudWatch Alarms](#) (p. 38)

### Create an Auto Scaling Group

First, create an Auto Scaling group, `my-asg`.

#### To create Auto Scaling group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**.
3. On the **Launch Configurations** page, select a launch configuration or create a new one. For more information, see [Create a Launch Configuration](#) (p. 49) or [Create an Auto Scaling Group](#) (p. 55).
4. Click **Create Auto Scaling group**.
5. On the **Configure Auto Scaling group details** page:
  - a. In the **Group name** field, enter a name for your Auto Scaling group (for example, `my-asg`).
  - b. In the **Group size** field, enter the number of instances you want your Auto Scaling group to start with (for example, 2).
  - c. In the **Network** field, select a VPC. If your AWS account supports EC2-Classic and the instance type specified by the launch configuration does not require a VPC, you can select EC2-Classic. For more information, see [Auto Scaling and Amazon Virtual Private Cloud](#) (p. 58).
  - d. Click the **Availability Zone(s)** field, and select an Availability Zone (for example, `us-west-2a`).
  - e. Click **Next: Configure scaling policies**.



1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

### Create Auto Scaling Group

Launch Configuration ⓘ my-test-ic

Group name ⓘ my-test-asg

Group size ⓘ Start with 2 instances

Network ⓘ Launch into EC2-Classic ⓘ Ⓢ Create new VPC

Availability Zone(s) ⓘ us-west-2a ✕

▸ Advanced Details

Cancel Next: Configure scaling policies

## Create Scaling Policies

Next, create two scaling policies: *my-scaleout-policy*, which increases the capacity of the group by 30 percent, and *my-scalein-policy*, which decreases the capacity of the group to two instances.

If you have already created CloudWatch alarms, you can associate them with the policies. If you have not created the alarms, you can create the alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies.

### To create scaling policies

1. On the **Configure scaling policies** page, select **Use scaling policies to adjust the capacity of this group**.
2. In the **Scale between** fields, enter the minimum size **1** and maximum size **5** instances for your Auto Scaling group.
3. In the **Increase Group Size** pane, enter the following information:
  - a. In the **Name:** field, enter a name for your policy, *my-scaleout-policy*.
  - b. If you have already created a CloudWatch alarm and want to associate it with the policy to increase group size, click the **Execute policy when** field and select your alarm.

If you have not created a CloudWatch alarm, click **Add new alarm** and complete the following steps to create a CloudWatch alarm.

- i. In the **Create Alarm** field, set the criteria for your alarm. In this example, set an alarm if the instance's average CPU utilization is above 80 percent.
- ii. The check box next to **Send a notification to** is selected by default. Select an existing topic, or click **Create topic** and enter a name. (Notifications use Amazon SNS.)
- iii. In the **With these recipients:** field, enter the email addresses of the recipients to notify. You can enter up to 10 email addresses, each separated by a comma. If you have entered your email address, check your email to confirm that you received the message from the topic.
- iv. Configure the threshold for your alarm by entering the following values:
  - In the **Whenever** fields, select **Average** and **CPU Utilization**.
  - In the **Is** fields, define the threshold for the alarm by selecting **>=** and entering **80**.
  - In the **For at least** fields, specify the sampling period and number of samples evaluated by the alarm. A shorter period creates a more sensitive alarm, while a longer period can mitigate brief spikes in a metric. For this example, monitor for 1 period of 5 minutes.

- In **Name of alarm**, a name is automatically generated for you. You can specify a different name now, but not after you create the alarm.

The screenshot shows the 'Create Alarm' window. It includes a checkbox for 'Send a notification to' with a value of 'AddCapacityNotification'. Below it is a text field for 'With these recipients' containing 'mymail@example.com'. The 'Whenever' section is set to 'Average' of 'CPU Utilization'. The 'Is' section is set to '>=' with a value of '80' and the unit 'Percent'. The 'For at least' section is set to '1' consecutive period(s) of '5 Minutes'. The 'Name of alarm' field contains 'AddCapacityAlarm'. On the right, there is a line graph titled 'CPU Utilization Percent' showing a blue line for 'my-test-asg' over a time period from 12/10 02:00 to 06:00. At the bottom right are 'Cancel' and 'Create Alarm' buttons.

v. Click **Create Alarm**.

- c. In the **Take the action:** fields, select Add, enter **30**, click and select Percent of Group, and enter **2**.
- d. Leave the **And then wait:** field set to the default value for this procedure.

4. In the **Decrease Group Size** pane

- a. In the **Name:** field, enter a name for your policy, **my-scale-in-policy**.
- b. If you have already created a CloudWatch alarm and want it to associate it with the policy to decrease group size, click the **Execute policy when** field and select your alarm.

If you have not created a CloudWatch alarm, click **Add new alarm** and complete the following steps to create a CloudWatch alarm.

- i. Follow the instructions in step 3b for setting up your notifications and then complete the following steps for configuring the threshold for your alarm:
  - In the **Whenever** fields, select **Average** and **CPU Utilization**.
  - In the **Is** fields, define the threshold for the alarm by selecting **<=** and entering **40**.
  - In the **For at least** fields, specify the sampling period and number of samples evaluated by the alarm. A shorter period creates a more sensitive alarm, while a longer period can mitigate brief spikes in a metric. For this example, monitor for 1 period of 5 minutes.
  - In the **Name of alarm** field, a name is automatically generated for you. You can change the name by typing in a new name now; you can't change the name after you create the alarm.
- ii. In the **Take the action:** field, select Remove, enter **2**, click and select instances.
- iii. Leave the **Wait** field set to the default value for this procedure.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Review

Create Auto Scaling Group

**Increase Group Size**

Name: my-scaleout-policy

Execute policy when: AddCapacityAlarm Edit Remove  
The alarm threshold is breached when CPUUtilization >= 80 for 300 seconds for these metric dimensions: AutoScalingGroupName = my-test-asg

Take the action: Add 30 Percent of group In 2 increments of at least

And then wait: 300 seconds before allowing another scaling activity

**Decrease Group Size**

Name: my-scalein-policy

Execute policy when: RemoveCapacityAlarm Edit Remove  
The alarm threshold is breached when CPUUtilization <= 40 for 300 seconds for these metric dimensions: AutoScalingGroupName = my-test-asg

Take the action: Remove 2 instances

And then wait: 300 seconds before allowing another scaling activity

Cancel Previous Review Next: Configure Notifications

5. Click **Review**.

### Verify Your Scaling Policies and CloudWatch Alarms

In this procedure, you verify that your scaling policies and the CloudWatch alarms are created for your Auto Scaling group.

#### To verify that scaling policies and the CloudWatch alarms are created

1. On the **Review** page, make any changes that you need and then click **Create Auto Scaling group**.
2. The **Auto Scaling Group creation status** page confirms that your Auto Scaling group was successfully created. Click **View your Auto Scaling Groups**.
3. On the **Auto Scaling Groups** page, select **my-asg**, the Auto Scaling group you just created. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab.
4. Click the **Scaling History** tab. The **Status** column lets you know that your Auto Scaling group has successfully launched 2 instances.
5. Click the **Instances** tab to view the current **Lifecycle** state of your newly launched instances. It takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to **InService**.

You can see that your Auto Scaling group has launched 2 new instances, and it is in **InService** lifecycle state. The **Health Status** column shows the result of the EC2 instance health check on your instance.

6. Click the **Scaling Policies** tab to can view the description of the policies created for the Auto Scaling group.

### Scaling with Metrics Using the AWS CLI

Use the AWS CLI as follows to create a launch configuration, Auto Scaling group, and Auto Scaling policies.

#### Tasks

- [Create a Launch Configuration \(p. 39\)](#)
- [Create an Auto Scaling Group \(p. 39\)](#)
- [Create Scaling Policies \(p. 39\)](#)

- [Create CloudWatch Alarms \(p. 40\)](#)

## Create a Launch Configuration

Use the following [create-launch-configuration](#) command to create a launch configuration named `my-lc`:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc
--image-id ami-514ac838 --instance-type m1.small --associate-public-ip-address
```

## 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 configuration `my-lc` that you just created:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg -
-launch-configuration-name my-lc --max-size 5 --min-size 1 --availability-zones
"us-east-1e"
```

## Create Scaling Policies

You can create scaling policies that tell the Auto Scaling group what to do when the specified conditions change.

### Example: my-scaleout-policy

Use the following [put-scaling-policy](#) command to create a scaling policy named `my-scaleout-policy` with an adjustment type of `PercentChangeInCapacity` that increases the capacity of the group by 30 percent:

```
aws autoscaling put-scaling-policy --policy-name my-scaleout-policy --auto-
scaling-group-name my-asg --scaling-adjustment 30 --adjustment-type Percent
ChangeInCapacity
```

Auto Scaling returns the ARN that serves as a unique name for the policy. Subsequently, you can use either the ARN or a combination of the policy name and group name to specify the policy. Store this ARN in a safe place. You'll need it to create CloudWatch alarms.

```
{
  "PolicyARN": "arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policy
Name/my-scaleout-policy
}
```

### Example: my-scalein-policy

Use the following [put-scaling-policy](#) command to create a scaling policy named `my-scalein-policy` with an adjustment type of `ChangeInCapacity` that decreases the capacity of the group by two instances:

```
aws autoscaling put-scaling-policy --policy-name my-scalein-policy --auto-
scaling-group-name my-asg --scaling-adjustment -2 --adjustment-type ChangeInCa
capacity
```

Auto Scaling returns the ARN for the policy. Store this ARN in a safe place. You'll need it to create CloudWatch alarms.

```
{
  "PolicyARN": "arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/my-asg:policy
Name/my-scalein-policy
}
```

## Create CloudWatch Alarms

In the previous task, you created scaling policies that provided instructions to the Auto Scaling group about how to scale in and scale out when the conditions that you specify change. In this task you create alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies.

### Example: AddCapacity

Use the following CloudWatch command [put-metric-alarm](#) to create an alarm named `AddCapacity` that increases the size of the Auto Scaling group when the average CPU usage of all the instances (CPUUtilization) increases to 80 percent (GreaterThanOrEqualToThreshold):

```
aws cloudwatch put-metric-alarm --alarm-name AddCapacity --metric-name
CPUUtilization --namespace AWS/EC2
--statistic Average --period 120 --threshold 80 --comparison-operator Greater
ThanOrEqualToThreshold
--dimensions "AutoScalingGroupName=my-asg" --evaluation-periods 2
--alarm-actions arn:aws:autoscaling...scalingPolicy:ac542982-cbeb-4294-891c-
a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-scaleout-policy
```

### Example: RemoveCapacity

Use the following CloudWatch command [put-metric-alarm](#) to create an alarm named `RemoveCapacity` that decreases the size of the Auto Scaling group when the average CPU usage of all the instances (CPUUtilization) decreases to 40 percent (LessThanOrEqualToThreshold):

```
aws cloudwatch put-metric-alarm --alarm-name RemoveCapacity --metric-name
CPUUtilization --namespace AWS/EC2
--statistic Average --period 120 --threshold 40 --comparison-operator
LessThanOrEqualToThreshold
--dimensions "Name=AutoScalingGroupName,Value=my-asg" --evaluation-periods 2

--alarm-actions arn:aws:autoscaling...scalingPolicy:4ee9e543-86b5-4121-b53b-
aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-scalein-policy
```

## Scheduled Scaling

Scaling based on a schedule allows you to scale your application in response to predictable load changes. For example, every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling activities based on the predictable traffic patterns of your web application.

To configure your Auto Scaling group to scale based on a schedule, you need to create scheduled actions. A scheduled action tells Auto Scaling to perform a scaling action at certain time in future. To create a scheduled scaling action, you specify the start time at which you want the scaling action to take effect, and you specify the new minimum, maximum, and desired size you want for that group at that time. At the specified time, Auto Scaling updates the group to set the new values for minimum, maximum, and desired sizes, as specified by your scaling action.

You can create scheduled actions for scaling one time only or for scaling on a recurring schedule.

### Contents

- [Programming Considerations for Scheduled Actions \(p. 41\)](#)
- [Scheduling Scaling Using the AWS CLI \(p. 41\)](#)

## Programming Considerations for Scheduled Actions

When you create a scheduled action, keep the following programming considerations in mind.

- Auto Scaling guarantees the order of execution for scheduled actions within the same group, but not for scheduled actions across groups.
- A scheduled action generally executes within seconds. However, the action may be delayed for up to two minutes from the scheduled start time. Because Auto Scaling executes actions within an Auto Scaling group in the order they are specified, scheduled actions with scheduled start times close to each other may take longer to execute.
- You can schedule a scheduled action for up to a month in the future.
- You can create a maximum of 125 scheduled actions per month per Auto Scaling group. This allows scaling four times a day for a 31-day month for each Auto Scaling group.
- A scheduled action must have a unique time value. If you attempt to schedule an activity at a time when another existing activity is already scheduled, the call is rejected with an error message noting the conflict.

## Scheduling Scaling Using the AWS CLI

Complete the following tasks to create a scheduled action to scale your Auto Scaling group.

### Tasks

- [Create a Launch Configuration \(p. 41\)](#)
- [Create an Auto Scaling Group \(p. 41\)](#)
- [Create a Schedule for Scaling Actions \(p. 41\)](#)

### Create a Launch Configuration

Use the following `create-launch-configuration` command to create a launch configuration named `my-lc`:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc --image-id ami-514ac838 --instance-type m1.small --associate-public-ip-address
```

### Create an Auto Scaling Group

Use the following `create-auto-scaling-group` command to create an Auto Scaling group named `my-asg`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --launch-configuration-name my-lc --max-size 5 --min-size 1 --availability-zones "us-east-1e"
```

### Create a Schedule for Scaling Actions

You can create a schedule for scaling one time only or for scaling on a recurring schedule.

#### To schedule scaling for one time only

To increase the number of running instances in your Auto Scaling group at a specific time, use the following [put-scheduled-update-group-action](#) command to create a scheduled action named `ScaleUp` that runs at the specified time (specified in "YYYY-MM-DDThh:mm:ssZ" format in UTC time):

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name ScaleUp
--auto-scaling-group-name my-asg --start-time "2013-05-12T08:00:00Z" --desired-
capacity 3
```

To decrease the number of running instances in your Auto Scaling group at a specific time, use the following `as-put-scheduled-update-group-action` command to create a scheduled action named `ScaleDown` that runs at the specified time (specified in "YYYY-MM-DDThh:mm:ssZ" format in UTC time):

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name
ScaleDown --auto-scaling-group-name my-asg --start-time "2013-05-13T08:00:00Z"
--desired-capacity 1
```

### To schedule scaling on a recurring schedule

You can specify a recurrence schedule using the Unix cron syntax format. For more information about cron syntax, see the [Cron](#) Wikipedia entry.

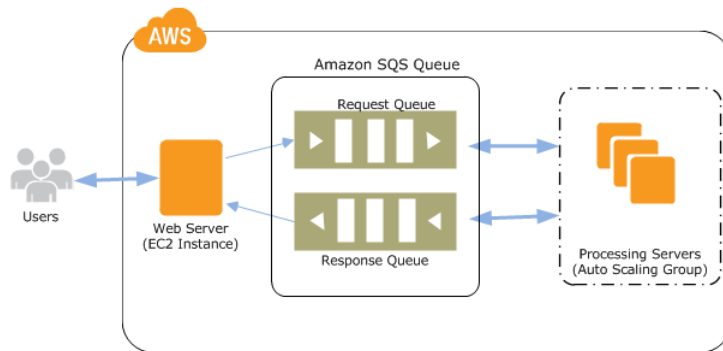
Use the following [put-scheduled-update-group-action](#) command to create a scheduled action named `scaleup-schedule-year` that runs at 00:30 hours on the first of January, June, and December each year:

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name
scaleup-schedule-year --auto-scaling-group-name my-asg --recurrence "30 0 1
1,6,12 0" --desired-capacity 3
```

## Scaling Based on Amazon SQS

Amazon Simple Queue Service (Amazon SQS) is a scalable message queuing system that stores messages as they travel between various components of your application architecture. Amazon SQS enables web service applications to quickly and reliably queue messages that are generated by one component and consumed by another component. A queue is a temporary repository for messages that are awaiting processing. For more information about Amazon SQS, see the [Amazon Simple Queue Service Developer Guide](#).

For example, suppose that you have a web app that receives orders from customers. The app runs on EC instances in an Auto Scaling group that is configured to handle a typical amount of orders. The app places the orders in an Amazon SQS queue until they are picked up for processing, processes the orders, and then sends the processed orders back to the customer. The following diagram illustrates the architecture of this example.



This architecture works well if your order levels remain the same at all times. What happens if your order levels change? You would need to launch additional EC2 instances when the orders increase and terminate the extra EC2 instances when the orders decrease. If your orders increase and decrease on a predictable schedule, you can specify the time and date to perform scaling activities. For more information, see [Scheduled Scaling](#) (p. 40). Otherwise, you can scale based on criteria, such as the number of messages in your Amazon SQS queue. For more information, see [Dynamic Scaling](#) (p. 32).

Queues provide a convenient mechanism to determine the load on an application. You can use the length of the queue (number of messages available for retrieval from the queue) to determine the load. Because each message in the queue represents a request from a user, measuring the length of the queue is a fair approximation of the load on the application. CloudWatch integrates with Amazon SQS to collect, view, and analyze metrics from Amazon SQS queues. You can use the metrics sent by Amazon SQS to determine the length of the Amazon SQS queue at any point in time. For a list of all the metrics that Amazon SQS sends to CloudWatch, see [Amazon SQS Metrics](#) in the *Amazon Simple Queue Service Developer Guide*.

The following examples create Auto Scaling policies that configure your Auto Scaling group to scale based on the number of messages in your Amazon SQS queue.

## Scaling with Amazon SQS Using the AWS CLI

The following example shows you how to create policies for scaling in and scaling out, plus create, verify, and validate CloudWatch alarms for your scaling policies. It assumes that you already have an Amazon SQS queue, an Auto Scaling group, and EC2 instances running the application that uses the Amazon SQS queue.

### Create the Scaling Policies

You can create scaling policies that tell the Auto Scaling group what to do when the specified conditions change.

#### To create scaling policies

1. Use the following [put-scaling-policy](#) command to create a scale out policy to increase the Auto Scaling group by one EC2 instance:

```
aws autoscaling put-scaling-policy --policy-name my-sqs-scaleout-policy -  
--auto-scaling-group-name my-asg --scaling-adjustment 1 --adjustment-type  
ChangeInCapacity
```

Auto Scaling returns the Amazon Resource Name (ARN) for the new policy. Store the ARN in a safe place. You'll need it when you create the CloudWatch alarms.

2. Use the following [put-scaling-policy](#) command to create a scale in policy to decrease the Auto Scaling group by one EC2 instance:



```
aws autoscaling put-scaling-policy --policy-name my-sqs-scalein-policy -  
--auto-scaling-group-name my-asg --scaling-adjustment -1 --adjustment-type  
ChangeInCapacity
```

Auto Scaling returns the ARN for the new policy. Store the ARN in a safe place. You'll need it when you create the CloudWatch alarms.

## Create the CloudWatch Alarms

Next, you create alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies that you created in the previous task.

### Note

All active Amazon SQS queues send metrics to CloudWatch every five minutes. We recommend that you set the alarm `Period` to at least 300 seconds. Setting the alarm `Period` to less than 300 seconds will result in alarm going to `INSUFFICIENT_DATA` state while waiting for the metrics.

## To create CloudWatch alarms

1. Use the following `put-metric-alarm` command to create an alarm that increases the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) increases to three and remains at three or greater for at least five minutes.

```
aws cloudwatch put-metric-alarm --alarm-name AddCapacityToProcessQueue -  
--metric-name ApproximateNumberOfMessagesVisible --namespace "AWS/SQS"  
--statistic Average --period 300 --threshold 3 --comparison-operator Great  
erThanOrEqualToThreshold --dimensions Name=QueueName,Value=my-queue  
--evaluation-periods 2 --alarm-actions arn
```

2. Use the following `put-metric-alarm` command to create an alarm that decreases the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) decreases to one and the length remains at one or fewer for at least five minutes.

```
aws cloudwatch put-metric-alarm --alarm-name RemoveCapacityFromProcessQueue  
--metric-name ApproximateNumberOfMessagesVisible --namespace "AWS/SQS"  
--statistic Average --period 300 --threshold 1 --comparison-operator  
LessThanOrEqualToThreshold --dimensions Name=QueueName,Value=my-queue  
--evaluation-periods 2 --alarm-actions arn
```

## Verify Your Scaling Policies and CloudWatch Alarms

You can verify that your CloudWatch alarms and scaling policies were created.

### To verify your CloudWatch alarms

Use the following `describe-alarms` command:

```
aws cloudwatch describe-alarms --alarm-names AddCapacityToProcessQueue Remove  
CapacityFromProcessQueue
```

### To verify your scaling policies

Use the following [describe-policies](#) command:

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

### Test Your Scale Out and Scale In Policies

You can test your scale out policy by increasing the number of messages in your Amazon SQS queue and then verifying that your Auto Scaling group has launched an additional EC2 instance. Similarly, you can test your scale in policy by decreasing the number of messages in your Amazon SQS queue and then verifying that the Auto Scaling group has terminated an EC2 instance.

#### To test the scale out policy

1. Follow the steps in [Sending a Message](#) to add messages to your Amazon SQS queue. Make sure that you have at least three messages in the queue.

It takes a few minutes for the Amazon SQS queue metric `ApproximateNumberOfMessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to launch one EC2 instance.

2. Use the following [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 policy

1. Follow the steps in [Deleting a Message](#) to remove messages from the Amazon SQS queue. Make sure that you have no more than one message in the queue.

It takes a few minutes for the Amazon SQS queue metric `ApproximateNumberOfMessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to terminate one EC2 instance.

2. Use the following [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
```

## Controlling Access to Your Auto Scaling Resources

Auto Scaling integrates with AWS Identity and Access Management (IAM), a service that enables you to do the following:

- Create users and groups under your organization's AWS account
- Assign unique security credentials to each user under your AWS account
- Control each user's permissions to perform tasks using AWS resources
- Allow the users in another AWS account to share your AWS resources

- Create roles for your AWS account and define the users or services that can assume them
- Use existing identities for your enterprise to grant permissions to perform tasks using AWS resources

For example, you could create an IAM policy that grants the Managers group permission to use only `DescribeAutoScalingGroups`, `DescribeLaunchConfigurations`, `DescribeScalingActivities`, and `DescribePolicies`. Users in the Managers group could then use those actions with any Auto Scaling groups and launch configurations. Note that you can't restrict access to a particular Auto Scaling group or launch configuration.

For more information about IAM, see the following:

- [Identity and Access Management \(IAM\)](#)
- [IAM Getting Started Guide](#)
- [Using IAM](#)

#### Contents

- [Auto Scaling Actions](#) (p. 46)
- [Auto Scaling Resources](#) (p. 46)
- [Auto Scaling Keys](#) (p. 46)
- [Example IAM Policies for Auto Scaling](#) (p. 47)
- [Launch Auto Scaling Instances with an IAM Role](#) (p. 48)

## Auto Scaling Actions

In an IAM policy, you can specify any and all Auto Scaling actions. For Auto Scaling, use the following prefix with the name of the action: `autoscaling:`. For example: `autoscaling:CreateAutoScalingGroup` and `autoscaling:CreateLaunchConfiguration`. You can also use wildcards. For example, use `autoscaling:*` to indicate all Auto Scaling actions.

For a list of the Auto Scaling actions, see [Auto Scaling Actions](#) in the *Auto Scaling API Reference*.

## Auto Scaling Resources

When writing an IAM policy to control access to Auto Scaling actions, you must use `""` as the resource. There are no supported Amazon Resource Names (ARNs) for Auto Scaling resources.

## Auto Scaling Keys

Auto Scaling implements the following policy keys only.

#### AWS-Wide Policy Keys

- `aws:CurrentTime`—To check for date/time conditions.
- `aws:EpochTime`—To check for date/time conditions using a date in epoch or UNIX time.
- `aws:principaltype`—To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`—To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

- `aws:SourceArn`—To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see [Amazon Resource Name \(ARN\)](#) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`—To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.
- `aws:UserAgent`—To check the client application that made the request.
- `aws:userid`—To check the user ID of the requester.
- `aws:username`—To check the user name of the requester, if available.

**Note**

Key names are case sensitive.

## Example IAM Policies for Auto Scaling

The following are simple IAM policies that you can use to control user access to Auto Scaling. The resource is always `"*"`, because you can't specify a particular Auto Scaling resource in a policy.

### Example 1: Create and manage Auto Scaling launch configurations

The following policy grants users permission to use all Auto Scaling actions that include the string `LaunchConfiguration` in their names.

Alternatively, you can list each action explicitly instead of using wildcards. If you list each action separately, the policy would not automatically apply to any new Auto Scaling actions we introduce that include the string `LaunchConfiguration` in their names.

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

### Example 2: Create and manage Auto Scaling groups and policies.

The following policy grants users permission to use all Auto Scaling actions that include the string `Scaling` in their names.

Alternatively, you can list each action explicitly instead of using wildcards. If you list each action separately, the policy would not automatically apply to any new Auto Scaling actions we introduce that include the string `Scaling` in their names.

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

### Example 3: Change the capacity of Auto Scaling groups.

The following policy grants users permission to use the `SetDesiredCapacity` action to change the capacity of Auto Scaling groups.

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

## Launch Auto Scaling Instances with an IAM Role

AWS Identity and Access Management (IAM) roles for EC2 instances make it easier for you to access other AWS services securely from within the EC2 instances. EC2 instances launched with an IAM role automatically have AWS security credentials available.

You can use IAM roles with Auto Scaling to automatically enable applications running on your EC2 instances to securely access other AWS resources.

To launch EC2 instances with an IAM role in Auto Scaling, you'll have to create an Auto Scaling launch configuration with an EC2 instance profile. An instance profile is simply a container for an IAM role. First, create an IAM role that has all the permissions required to access the AWS resources, then add your role to the instance profile.

For more information about IAM roles and instance profiles, see [Delegating API Access by Using Roles](#) in the *Using IAM* guide.

## Prerequisites: Using IAM

Use these steps for launching Auto Scaling instances with an IAM role. Before you walk, be sure you've completed the following steps using IAM:

- Create an IAM role.
- Create an instance profile.
- Add the IAM role to the instance profile.
- Retrieve the name of the instance profile or the full Amazon Resource Name (ARN) of the instance profile.

For more information about creating and managing an IAM role, see [Create a Role](#) in the *Using IAM* guide.

## Create a Launch Configuration

When you create the launch configuration, specify the name of the instance profile or the full ARN of the instance profile.

For example, use the following [create-launch-configuration](#) command:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-with-instance-profile --image-id ami-baba68d3 --instance-type m1.small --iam-instance-profile my-instance-profile
```

## Create an Auto Scaling Group

Create your Auto Scaling group, specifying the launch configuration that you just created.

For example, use the following `create-auto-scaling-group` command:

```
aws autoscaling create-auto-scaling-group --launch-configuration-name my-asg-with-instance-profile --launch-configuration my-lc-with-instance-profile --availability-zones "us-east-1e" --max-size 1 --min-size 1
```

# Creating Launch Configurations

A *launch configuration* is a template for the EC2 instances launched into an Auto Scaling group. You must specify a launch configuration when you create an Auto Scaling group. You can't modify a launch configuration after you've created it. However, you can change which launch configuration is associated with an Auto Scaling group at any time.

If you've launched an EC2 instance before, you've already walked through the process of defining compute characteristics such as the instance type, security groups, and configuration scripts. You define these same characteristics for any instances launched into the Auto Scaling group using a launch configuration.

When your launch configuration is ready, you can move straight to [creating your Auto Scaling group](#) (p. 54). However, you might also want to look at how to [control when your instances launch and terminate](#) (p. 62), [how to tag instances](#) (p. 73) so they're easier to identify, and [how to incorporate Spot Instances](#) (p. 76) to make your Auto Scaling group even more cost effective.

### Contents

- [Create a Launch Configuration](#) (p. 49)
- [Create a Launch Configuration Using an EC2 Instance](#) (p. 50)

## Create a Launch Configuration

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

### To create a launch configuration using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar at the top of the screen, the current region is displayed. Select a region for your Auto Scaling group that meets your needs.
3. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**. If you are new to Auto Scaling, you see a welcome page; click **Create Auto Scaling group**.
4. Click **Create launch configuration**.
5. On the **Choose AMI** page, select an AMI.
6. On the **Choose Instance Type** page, select a hardware configuration for your instance. Click **Next: Configure details**.

**Note**

T2 instances must be launched into a subnet of a VPC. If you select a `t2.micro` instance but don't have a VPC, one is created for you. This VPC includes a public subnet in each Availability Zone in the region.

7. On the **Configure Details** page, do the following:
  - a. In the **Name** field, enter a name for your launch configuration.
  - b. Under **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to you instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.
  - c. Click **Skip to review**.
8. On the **Review** page, click **Edit security groups**, follow the instructions to choose an existing security group, and then click **Review**.
9. On the **Review** page, click **Create launch configuration**.
10. In the **Select an existing key pair or create a new key pair** field, select one of the listed options. Click the acknowledgment check box, and then click **Create launch configuration**.

**Warning**

Do not select **Proceed without a key pair** if you need to connect to your instance.

## Create a Launch Configuration Using an EC2 Instance

Auto Scaling provides you with an option to create a new launch configuration using the attributes from a running EC2 instance. When you use this option, Auto Scaling copies the attributes from the specified instance into a template from which you can launch one or more Auto Scaling groups.

**Tip**

You can [create an Auto Scaling group directly from an EC2 instance \(p. 56\)](#). When you use this feature, Auto Scaling automatically creates a launch configuration for you as well.

If the specified instance has properties that are not currently supported by Auto Scaling, instances launched by Auto Scaling using the launch configuration created from the identified instance might not be identical to the identified 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, by default Auto Scaling derives attributes for the launch configuration from the specified instance, plus the block device mapping for the AMI that the instance was launched from (ignoring any additional block devices that were added to the instance 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, monitoring, placement tenancy, ramdisk, security groups, Spot Price, user data, whether the instance has a public IP address is associated, and whether the instance is EBS-optimized.

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

**Examples**

- [Create a Launch Configuration Using an EC2 Instance \(p. 51\)](#)

- [Create a Launch Configuration from an Instance and Override the Block Devices](#) (p. 52)
- [Create a Launch Configuration and Override the Instance Type](#) (p. 53)

## 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 Using the AWS Management 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. 95).

## Create a Launch Configuration from an EC2 Instance Using the AWS CLI

Use the following [create-launch-configuration](#) command to create a new launch configuration from an instance using the same attributes as the instance (other than any block devices added after launch, which 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

By default, Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration, except that 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 Using the AWS CLI

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\":{\"Snapshot  
Id\":\"snap-3decf207\"}}, {\"DeviceName\":\"/dev/sdf\", \"Ebs\":{\"Snapshot  
Id\":\"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-ee6ac86"  
      }  
    }  
  ],  
  "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

By default, Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration. Depending on your requirements, you might want to change some of these attributes. Auto Scaling provides you with options 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 Using the AWS CLI

Use the following [create-launch-configuration](#) command to create a launch configuration using an EC2 instance but with a different instance type (for example `m1.small`) than the instance (for example `t1.micro`):

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-changetype --instance-id i-a8e09d9c --instance-type m1.small
```

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": "m1.small",
    "AssociatePublicIpAddress": true
  }
}
```

## Creating Auto Scaling Groups

An Auto Scaling group is a collection of EC2 instances managed by the Auto Scaling service. Each Auto Scaling group contains configuration options that control when Auto Scaling should launch new instances and terminate existing instances. At a minimum, an Auto Scaling group must contain the following:

- A name
- The maximum number of instances that can be in the Auto Scaling group
- The minimum number of instances that can be in the Auto Scaling group

However, an Auto Scaling group with these options only does not provide much value. The following are additional configuration options that you should define to get the most out of your Auto Scaling group:

- Desired capacity. This parameter specifies the number of instances you'd like to have in the Auto Scaling group.
- Availability Zones or subnets. It is often a good idea to build or modify your applications in AWS to use more than one Availability Zone. If your Auto Scaling group operates within a VPC, you can alternatively specify which subnets you want Auto Scaling to use.
- Launch configuration. As described in [Creating Launch Configurations \(p. 49\)](#), you must define an instance type and how each instance will be configured.
- Metrics and health checks. An effective Auto Scaling group uses metrics to determine when it should launch or terminate instances. In addition, it's helpful to define health checks which Auto Scaling uses to determine if an instance is healthy or, if not, if Auto Scaling should terminate the instance and replace it.

After you create your Auto Scaling, read [Configuring Your Auto Scaling Groups \(p. 85\)](#) to learn about actions that you can take and [Monitoring Your Auto Scaling Instances \(p. 114\)](#) to learn about tracking the performances of instances in the Auto Scaling group.

### Contents

- [Create an Auto Scaling Group \(p. 55\)](#)

- [Create an Auto Scaling Group from an EC2 Instance \(p. 56\)](#)

## Create an Auto Scaling Group

When you create an Auto Scaling group, you must specify the launch configuration to use for launching the instances, and the number of instances your group must maintain at all times. You can also specify the Availability Zone in which you want the instances to be launched.

### Prerequisites

Create a launch configuration. For more information, see [Create a Launch Configuration \(p. 49\)](#).

### To create an Auto Scaling group using the console

1. Open the Amazon EC2 console.
2. In the navigation bar at the top of the screen, the current region is displayed. Select the same region as the launch configuration.
3. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
4. Click **Create Auto Scaling group**.
5. On the **Create Auto Scaling Group** page, select **Create an Auto Scaling group from an existing launch configuration**, select a launch configuration, and then click **Next Step**.
6. On the **Configure Auto Scaling group details** page, do the following:
  - a. In **Group name**, enter a name for your Auto Scaling group.
  - b. In **Group size**, enter the desired capacity for your Auto Scaling group.
  - c. If you are launching a T2 instance, you must select a VPC in **Network**. Otherwise, if your account supports EC2-Classic and you are launching a type of instance that doesn't require a VPC, you can select either **Launch into EC2-Classic** or a VPC.
  - d. If you selected a VPC in the previous step, select a subnet from **Subnet**. If you selected EC2-Classic in the previous step, select an Availability Zone from **Availability Zone(s)**.
  - e. Click **Next: Configure scaling policies**.
7. In the **Configure scaling policies** page, select one of the following options, and then click **Review**:
  - To automatically adjust the size of the Auto Scaling group based on criteria that you specify, select **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see [Create Scaling Policies \(p. 36\)](#).
  - To manually adjust the size of the Auto Scaling group, select **Keep this group at its initial size**. For more information, see [Manual Scaling \(p. 30\)](#).
8. (Optional) To add tags now, click **Edit tags** and complete the following steps. Alternatively, you can add tags later on. For more information, see [Tagging Auto Scaling Groups and Instances \(p. 73\)](#).
  - a. In the **Key** and **Value** fields, enter the key and the value for your first tag.
  - b. Keep **Tag New Instances** selected if you want Auto Scaling to propagate the tag to the instances launched by your Auto Scaling group.
  - c. Click **Add tag** to add additional tags and then specify keys and values for the tags.
  - d. Click **Review**.
9. On the **Review** page, click **Create Auto Scaling group**.
10. On the **Auto Scaling group creation status** page, click **Close**.

## 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 from an EC2 Instance

Auto Scaling provides you with the option to create an Auto Scaling group by specifying an EC2 instance, instead of a launch configuration, and by specifying attributes such as the minimum, maximum, and desired number of EC2 instances for the Auto Scaling group.

When you create an Auto Scaling group using an EC2 instance, Auto Scaling automatically creates a launch configuration for you and associates it with the Auto Scaling group. This launch configuration has the same name as the Auto Scaling group, and it derives its attributes, such as AMI ID, instance type, and Availability Zone, from the specified instance.

## Limitations

The following are limitations when creating an Auto Scaling group from an EC2 instance:

- If the identified instance has tags, the tags are not copied to the `Tags` attribute of the new Auto Scaling group.
- The Auto Scaling group includes the block device mapping from the AMI used to launch the instance; it does not include any block devices attached after instance launch.
- If the identified instance is registered with one or more load balancers, the load balancer names are not copied to the `LoadBalancerNames` attribute of the new Auto Scaling group.

## Prerequisites

Before you begin, find the ID of the EC2 instance using the Amazon EC2 console or the [describe-instances](#) command (AWS CLI).

The EC2 instance must meet the following criteria:

- The instance is in the 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 `running` state.
- The AMI used to launch the instance must still exist.

## Contents

- [Create an Auto Scaling Group from an EC2 Instance Using the Console](#) (p. 57)
- [Create an Auto Scaling Group from an EC2 Instance Using the AWS CLI](#) (p. 57)

## Create an Auto Scaling Group from an EC2 Instance Using the Console

You can use the console to create 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. 95).

## Create an Auto Scaling Group from an EC2 Instance Using the AWS CLI

Use the following `create-auto-scaling-group` command to create an Auto Scaling group, `my-asg-from-instance`, from the EC2 instance `i-7f12e649`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance --instance-id i-7f12e649 --min-size 1 --max-size 2 --desired-capacity 2
```

```
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 also named `my-asg-from-instance`:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 0,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-asg-from-instance",
      "DefaultCooldown": 300,
      "MinSize": 1,
      "Instances": [
        {
          "InstanceId": "i-6bd79d87",
          "AvailabilityZone": "us-east-1a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "my-asg-from-instance"
        },
        {
          "InstanceId": "i-6cd79d80",
          "AvailabilityZone": "us-east-1a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "my-asg-from-instance"
        }
      ],
      "MaxSize": 2,
    }
  ]
}
```

```
        "VPCZoneIdentifier": "subnet-6bea5f06",
        "TerminationPolicies": [
            "Default"
        ],
        "LaunchConfigurationName": "my-asg-from-instance",
        "CreatedTime": "2014-12-29T16:14:50.397Z",
        "AvailabilityZones": [
            "us-east-1a"
        ],
        "HealthCheckType": "EC2"
    }
}
```

Use the following `describe-launch-configs` command to describe the launch configuration *my-asg-from-instance*.

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

## Auto Scaling and Amazon Virtual Private Cloud

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. For more information, see the [Amazon VPC User Guide](#).

Within a virtual private cloud (VPC), you can launch AWS resources such as an Auto Scaling group. An Auto Scaling group in a VPC works essentially the same way as it does on Amazon EC2 and supports the same set of features. This section provides you with an overview of Auto Scaling groups in a VPC and steps you through the process of creating an Auto Scaling group in a VPC. If you want to launch your Auto Scaling instances in Amazon EC2, see [Getting Started with Auto Scaling](#) (p. 16).

Before you can create your Auto Scaling group in a VPC, you must first configure your VPC environment. You create your VPC by specifying a range of IP addresses in the classless inter-domain routing (CIDR) range of your choice (for example, 10.0.0.0/16). For more information about CIDR notation and what "/16" means, go to [Classless Inter-Domain Routing](#) on Wikipedia.

You can create a VPC that spans multiple Availability Zones then add one or more subnets in each Availability Zone. 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.

### Contents

- [Default VPC](#) (p. 59)
- [IP Addressing in a VPC](#) (p. 59)
- [Instance Placement Tenancy](#) (p. 59)

- [Linking EC2-Classic Instances to a VPC \(p. 60\)](#)
- [Launch Auto Scaling Instances in a VPC \(p. 62\)](#)

## Default VPC

If you have created your AWS account after 2013-12-04 or you are creating your Auto Scaling group in a new region, we create a default VPC for you. Your default VPC comes with a subnet in each Availability Zone. If you have a default VPC, by default, your Auto Scaling group is created in the default VPC.

A default VPC combines the benefits of the advanced features provided by Amazon VPC platform with the ease of use of the Amazon EC2 platform. You can launch instances into your default VPC without needing to know anything about Amazon VPC.

For information about default VPC and to find out if your account comes with a default VPC, see [Your Default VPC and Subnets](#) in the *Amazon VPC Developer Guide*.

The steps for creating an Auto Scaling group in a default VPC is similar to the steps for creating an Auto Scaling group in Amazon EC2. If your AWS account comes with a default VPC and if you want to create your Auto Scaling group in a default VPC, follow the instructions in [Getting Started with Auto Scaling \(p. 16\)](#).

## IP Addressing in a VPC

When you launch your Auto Scaling instances in a VPC, your instances are automatically assigned with a private IP address in the address range of the subnet. This enables your instances to communicate with other instances in the VPC. You have an option to assign a public IP address to your instance. Assigning a public IP address to your instance allows it to communicate with the Internet or other services in AWS. You can choose the option of assigning public IP address to your instances when you create your launch configuration.

## Instance Placement Tenancy

Dedicated Instances are physically isolated at the host hardware level from instances that aren't dedicated and from instances that belong to other AWS accounts. 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, by default, they run as shared-tenancy instances. If you set the tenancy attribute of a VPC to `dedicated`, all instances launched in the VPC run as single-tenancy instances. For more information, see [Dedicated Instances](#) in the *Amazon VPC User Guide*. For pricing information, see the [Amazon EC2 Dedicated Instances](#) product page.

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. 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 instance	Dedicated Instance
default	shared-tenancy instance	Dedicated Instance
dedicated	Dedicated Instance	Dedicated Instance

You can specify the instance placement tenancy for your launch configuration as `default` or `dedicated` using the [create-launch-configuration](#) command with the `--placement-tenancy` option. For example, the following command 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. Note that `PlacementTenancy` is not part of the output for this command unless you have explicitly set the instance placement tenancy.

```
{
  "LaunchConfigurations": [
    {
      "UserData": null,
      "EbsOptimized": false,
      "PlacementTenancy": "dedicated",
      "LaunchConfigurationARN": "arn",
      "InstanceMonitoring": {
        "Enabled": true
      },
      "ImageId": "ami-b5a7ea85",
      "CreatedTime": "2015-03-08T23:39:49.011Z",
      "BlockDeviceMappings": [],
      "KeyName": null,
      "SecurityGroups": [],
      "LaunchConfigurationName": "my-launch-config",
      "KernelId": null,
      "RamdiskId": null,
      "InstanceType": "m3.medium"
    }
  ]
}
```

## Linking EC2-Classic Instances to a VPC

If you are launching the instances in your Auto Scaling group in EC2-Classic, you can link them to a VPC using *ClassicLink*. *ClassicLink* enables you to associate one or more security groups for the VPC with the EC2-Classic instances in your Auto Scaling group, enabling communication between these linked EC2-Classic instances and instances in the VPC using private IP addresses. For more information, see [ClassicLink](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you have running EC2-Classic instances in your Auto Scaling group, you can link them to a VPC with *ClassicLink* enabled. For more information, see [Linking an Instance to a VPC](#) in the *Amazon EC2 User Guide for Linux Instances*. Alternatively, you can update the Auto Scaling group to use a launch configuration that automatically links the EC2-Classic instances to a VPC at launch, then terminate the running instances and let Auto Scaling launch new instances that are linked to the VPC.

## Link to a VPC Using the AWS Management Console

Use the following procedure to create a launch configuration that links EC2-Classic instances to the specified VPC and update an existing Auto Scaling group to use the launch configuration.

### To link EC2-Classic instances in an Auto Scaling group to a VPC using the console

1. Verify that the VPC has ClassicLink enabled. For more information, see [Viewing Your ClassicLink-Enabled VPCs](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Create a security group for the VPC that you are going to link EC2-Classic instances to, with rules to control communication between the linked EC2-Classic instances and instances in the VPC.
3. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
4. In the navigation pane, click **Launch Configurations**. If you are new to Auto Scaling, you see a welcome page. Click **Create Auto Scaling group**.
5. Click **Create launch configuration**.
6. On the **Choose AMI** page, select an AMI.
7. On the **Choose an Instance Type** page, select an instance type, and then click **Next: Configure details**.
8. On the **Configure details** page, do the following:
  - a. Enter a name for your launch configuration.
  - b. Expand **Advanced Details**, select the **IP Address Type** that you need, and then select **Link to VPC**.
  - c. From **VPC**, select the VPC with ClassicLink enabled from step 1.
  - d. From **Security Groups**, select the security group from step 2.
  - e. Click **Skip to review**.
9. On the **Review** page, make any changes that you need, and then click **Create launch configuration**. In the **Select an existing key pair or create a new key pair** field, select an option, click the acknowledgment check box (if present), and then click **Create launch configuration**.
10. When prompted, follow the directions to create an Auto Scaling group that uses the new launch configuration. Be sure to select **Launch into EC2-Classic** for **Network**. Otherwise, click **Cancel** and then add your launch configuration to an existing Auto Scaling group as follows:
  - a. In the navigation pane, click **Auto Scaling Groups**.
  - b. Select your Auto Scaling group, click **Actions**, and then click **Edit**.
  - c. From **Launch Configuration**, select your new launch configuration and then click **Save**.

## Link to a VPC Using the AWS CLI

Use the following procedure to create a launch configuration that links EC2-Classic instances to the specified VPC and update an existing Auto Scaling group to use the launch configuration.

### To link EC2-Classic instances in an Auto Scaling group to a VPC using the AWS CLI

1. Verify that the VPC has ClassicLink enabled. For more information, see [Viewing Your ClassicLink-Enabled VPCs](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Create a security group for the VPC that you are going to link EC2-Classic instances to, with rules to control communication between the linked EC2-Classic instances and instances in the VPC.
3. Create a launch configuration using the [create-launch-configuration](#) command as follows, where *vpc\_id* is the ID of the VPC with ClassicLink enabled from step 1 and *group\_id* is the security group from step 2:

```
aws autoscaling create-launch-configuration --launch-configuration-name  
classiclink-config  
--image-id ami_id --instance-type instance_type  
--classic-link-vpc-id vpc_id --classic-link-vpc-groups group_id
```

4. Update your existing Auto Scaling group, for example *my-asg*, with the launch configuration that you created in the previous step. Any new EC2-Classical instances launched in this Auto Scaling group are linked EC2-Classical instances. Use the [update-auto-scaling-group](#) command as follows:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg  
--launch-configuration-name classiclink-config
```

Alternatively, you can use this launch configuration with a new Auto Scaling group that you create using [create-auto-scaling-group](#).

## Launch Auto Scaling Instances in a VPC

You can use Auto Scaling to launch instances into a virtual private cloud (VPC).

### Prerequisites

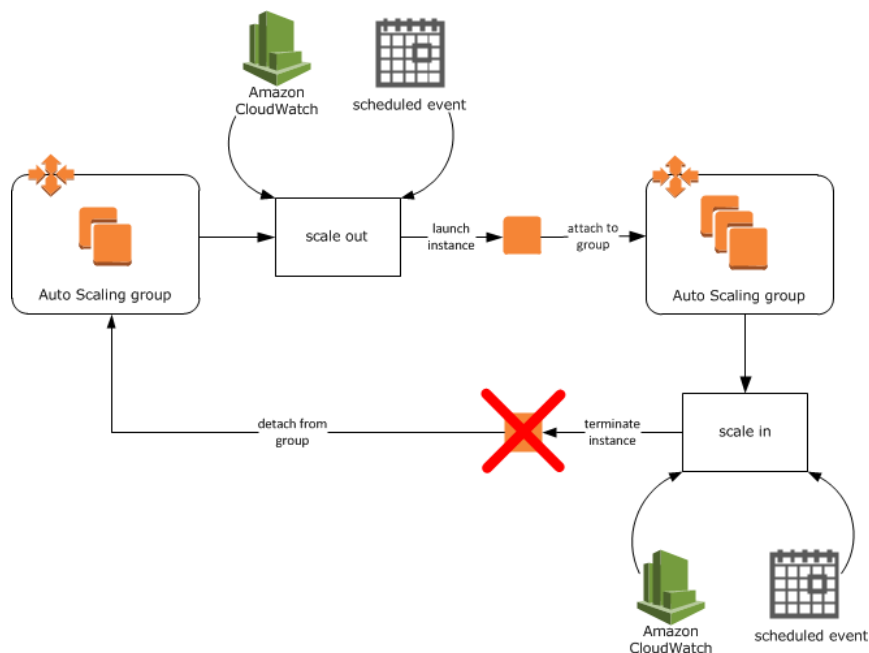
Before you can launch your Auto Scaling instances in a VPC, you must first create your VPC environment. After you create your VPC and subnets, you launch Auto Scaling instances within the subnets. The easiest way to create a VPC with one public subnet is to use the VPC wizard. For more information, see the [Amazon VPC Getting Started Guide](#).

### Examples

- [Getting Started with Auto Scaling \(p. 16\)](#)
- [Hosting a Web App on Amazon Web Services](#)
- [Hosting a .NET Web App on Amazon Web Services](#)

## Controlling How Instances Launch and Terminate

The section, [Auto Scaling Lifecycle \(p. 9\)](#), describes the basic lifecycle of instances as they launch or terminate within an Auto Scaling group.

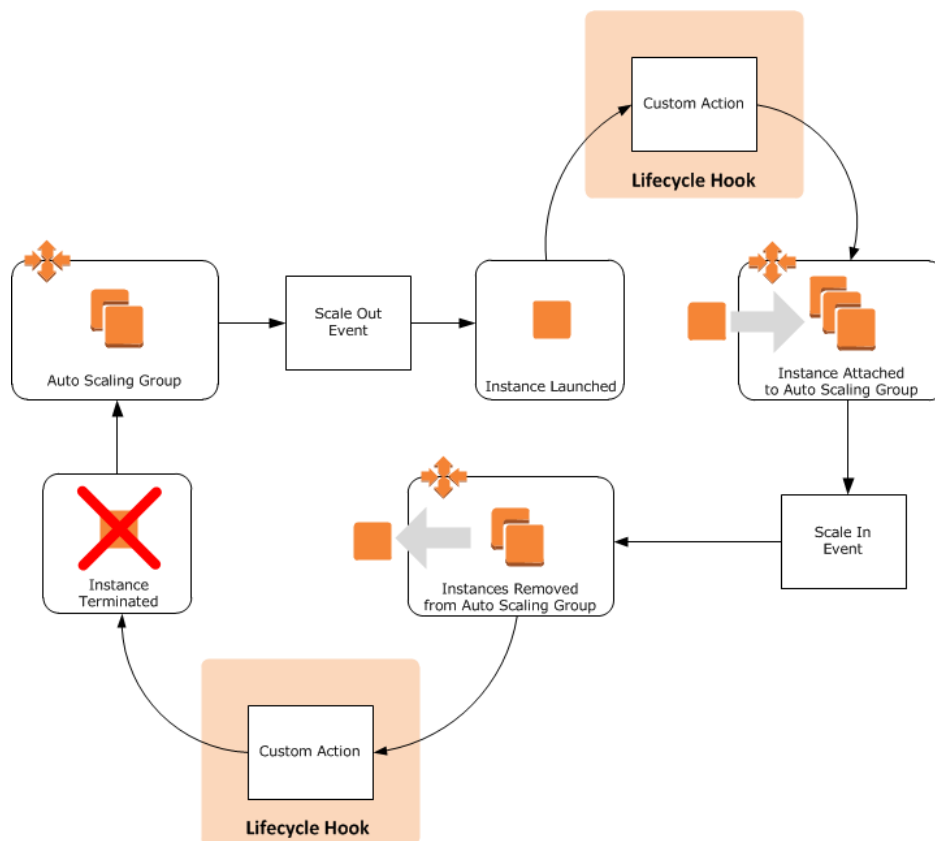


As this diagram illustrates, the standard procedure is for Auto Scaling to launch and configure an instance in response to a scale out event. When the instance is ready, it is immediately put into service. The same is true when a scale in event occurs. Auto Scaling selects an instances (based on any existing [termination policies](#) (p. 26) that are in place), removes it from the Auto Scaling group and terminates it.

While these processes are usually sufficient for most implementations of Auto Scaling groups, there are some situations in which you want more granular control over when instances are put into service and when they terminate. You can use *lifecycle hooks* to implement this level of control.

## Introducing Lifecycle Hooks

An Auto Scaling lifecycle hook allows you to add custom events to instances as they launch or terminate. A custom event could be actions such as manually installing software, or retrieving log files.



When you add a lifecycle hook to your Auto Scaling group:

1. Auto Scaling responds to a scale in or scale out event by launching or terminating an instance.
2. Auto Scaling puts the instance into a wait state. The state of the instance becomes either `Pending:Wait` or `Terminating:Wait`.
3. Auto Scaling sends a message to the notification target defined for the lifecycle hook. The message contains information about the instance that is launching or terminating, and a token you can use to control the lifecycle action.
4. At this point, the instance is ready for you to perform a custom action. The instance remains in a wait state until you tell Auto Scaling to continue or until the timeout period for the lifecycle hook ends.
5. By default, the instance remains in the `Pending:Wait` or `Terminating:Wait` state for one hour. If you take no action during that time, Auto Scaling terminates the instance. You can extend the length of time the instance remains in a waiting state by recording a heartbeat.

#### Note

You can only add lifecycle hooks to your Auto Scaling group through the CLI or API. There is no way to add a lifecycle hook using the AWS Management Console.

For more information about adding lifecycle hooks to your Auto Scaling groups, see the following:

- [Adding Lifecycle Hooks \(p. 65\)](#)
- [Considerations When Using Lifecycle Hooks \(p. 66\)](#)

## Adding Lifecycle Hooks

Each Auto Scaling can have multiple lifecycle hooks. However, you can have only a set number of hooks for each AWS account. For more information, see [Auto Scaling Account Limits](#).

### To add a lifecycle hook to an Auto Scaling group

1. Create a notification target. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

After you create your target, make a note of its Amazon Resource Name (ARN). For example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`.

2. Create an AWS Identity and Access Management role using the steps in [Creating a Role for an AWS Service \(AWS Management Console\)](#) in the *Using IAM* guide. When you are prompted to select a role type, choose **AWS Service Roles** and then select **AutoScaling Notification Access**.

After you create your role, make a note of its ARN. For example, `arn:aws:iam::123456789012:role/my-auto-scaling-role`.

3. Create a lifecycle hook, which tells Auto Scaling that you want to perform an action on the instance before it transitions. You create a lifecycle hook using the [put-lifecycle-hook](#) command as follows:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-lifecycle-hook
--auto-scaling-group-name my-asg --lifecycle-transition autoscaling:EC2_IN
STANCE_LAUNCHING --notification-target-arn sns-topic-arn --role-arn iam-
role-arn
```

Note that you can specify the `--heartbeat-timeout` parameter to determine how long Auto Scaling should keep an instance in the `Pending:Wait` or `Terminating:Wait` state.

4. Perform a custom action.
5. (Optional) If you need more time to complete the custom action, use the following [record-lifecycle-action-heartbeat](#) command to restart the heartbeat timeout and keep the instance in a waiting state. Note that the lifecycle action token is included in the message sent to your notification target.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-action-token
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635 --auto-scaling-group-name my-asg -
-lifecycle-hook-name my-lifecycle-hook
```

For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, if you discover that you need more time to complete your custom action, use the `record-lifecycle-action-heartbeat` command to restart the timeout value, giving you a total of 90 minutes to complete the custom action.

6. When you finish the custom action, let Auto Scaling know it can finish launching or terminating the instance. You use the [complete-lifecycle-action](#) command as follows:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-token bcd2f1b8-
9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-hook-name my-lifecycle-hook --auto-
scaling-group-name my-asg --lifecycle-action-result CONTINUE
```

## Considerations When Using Lifecycle Hooks

Adding lifecycle hooks to your Auto Scaling gives you a greater degree of control over how instances launch and terminate. Here are some things to consider when adding a lifecycle hook to your Auto Scaling, to help ensure that the group continues to perform as expected.

### Considerations

- [Keeping Instances in a Wait State](#) (p. 66)
- [Cooldowns and Custom Actions](#) (p. 66)
- [Abandon or Continue](#) (p. 66)
- [Spot Instances](#) (p. 67)

## Keeping Instances in a Wait State

Instances can only remain in a wait state for a finite period of time. The default length of time is 1 hour (3600 seconds). You can adjust this time in the following ways:

- Change the heartbeat timeout for the lifecycle hook. When you create a lifecycle hook, you can optionally define the timeout value. You accomplish this in the CLI with the `--heartbeat-timeout` parameter. In the API, use the `HeartbeatTimeout` parameter.
- Call the [complete-lifecycle-action](#) command or `CompleteLifecycleAction` action, which tells Auto Scaling that the instance is ready to continue to the next state.
- Call the [record-lifecycle-action-heartbeat](#) command or the `RecordLifecycleActionHeartbeat` action to increment the amount of time the instance remains in a wait state. The amount of time added is equal to the time assigned to the timeout value. For example, if the timeout value is 1 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.

### Note

You can only keep an instance in a wait state for a maximum of 48 hours, regardless of how often you call `record-lifecycle-action-heartbeat` or `RecordLifecycleActionHeartbeat`.

## Cooldowns and Custom Actions

Each time Auto Scaling launches or terminates an instance, a [cooldown](#) (p. 23) takes effect. This cooldown helps ensure that the Auto Scaling group does not launch or terminate more instances than needed.

When you put a lifecycle hook on an Auto Scaling group, any scaling actions are suspended until the instance move out of a wait state. After the instance moves out of a wait state, the cooldown period starts.

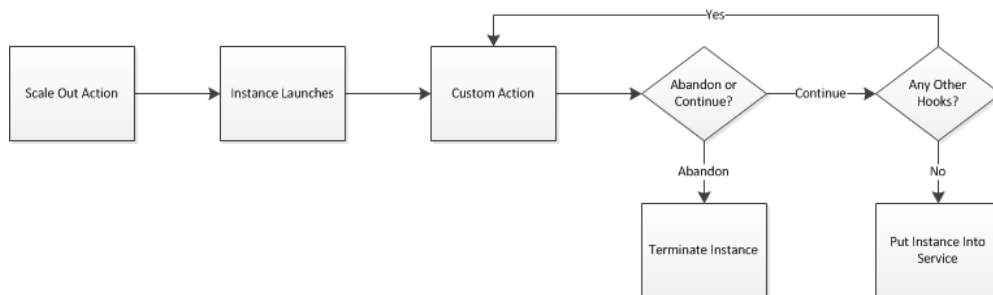
For example, consider an Auto Scaling group for a set of servers in a basic web application. This group has a lifecycle hook that allows for custom actions as a new instance launches. The application experiences an increase in demand, and Auto Scaling launches a new instance to address the need for additional capacity. Because there is a lifecycle hook, the instance is put into a `Pending:Wait` state, which means the instance is not available to handle traffic yet. Until the instance moves into service, all scaling actions are suspended for the Auto Scaling group. When the instance is put into service, the cooldown period starts and, when it expires, additional scaling actions can resume.

Note that if you add a lifecycle hook to perform actions as your instances launch, the health check grace period must allow enough time to complete the lifecycle hook.

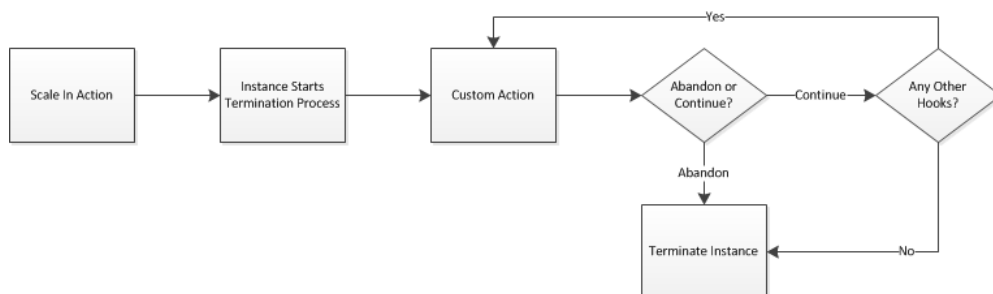
## Abandon or Continue

At the conclusion of a lifecycle hook, an instance can have one of two results: `ABANDON` or `CONTINUE`.

If the instance is launching, an `ABANDON` result means that whatever additional actions you wanted to take on the instance were unsuccessful. Instead of putting the instance into service, Auto Scaling terminates the instance and, if necessary, launches a new one. A `CONTINUE` result means that your actions were successful, and Auto Scaling can put the instance into service.



If the instance is terminating, an `ABANDON` result means stop any remaining actions, such as other lifecycle hooks, and move straight to terminating the instance. A `CONTINUE` result means continue with the termination process, but allow any other lifecycle hooks applied to the instance take effect as well.



#### Note

For terminating instances, both an `ABANDON` result and a `CONTINUE` result cause the instance to terminate. The main difference is whether any other actions are allowed to occur on the instance.

## Spot Instances

You can use lifecycle hooks with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating due to a change in the Spot Price, which can happen at any time. In addition, when a Spot Instance terminates, you must still complete the lifecycle action (using the **complete-lifecycle-action** command or the **CompleteLifecycleAction** action).

## Examples of How to Use Lifecycle Hooks

Lifecycle hooks can allow you to customize an Auto Scaling to meet the needs of your application's architecture. Here are some examples.

#### Examples

- [Installing Software to Pending Instances \(p. 68\)](#)
- [Filling a Cache of Servers \(p. 69\)](#)
- [Analyzing an Instance Before Termination \(p. 70\)](#)
- [Retrieving Logs from Terminating Instances \(p. 72\)](#)



## Installing Software to Pending Instances

As with a standalone EC2 instance, you have the option of configuring instances launched into an Auto Scaling group using user data. For example, you can specify a configuration script using the **User data** field in the AWS Management Console, or the `--userdata` parameter in the AWS CLI.

If you have software that can't be installed using a configuration script, or if you need to modify software manually before Auto Scaling adds the instance to the group, add a lifecycle hook to your Auto Scaling group that notifies you when the Auto Scaling group launches an instance. This hook keeps the instance in the `Pending:Wait` state while you install and configure the additional software.

By default, the instance remains in the `Pending:Wait` state for one hour. If you take no action during that time, Auto Scaling assumes that the instance was not configured correctly and terminates it. If you need more time, you can restart the timeout period. For example, if after 30 minutes you discover that you need more time to complete your software installation, you can restart the timeout period, giving you a total of 90 minutes to complete the software installation. If you are ready to add the instance to the Auto Scaling group before the timeout period ends, you can complete the lifecycle action.

## Adding Software Using the AWS CLI

The following steps demonstrate the general process for installing additional software on instances joining an Auto Scaling group.

### To add software manually to pending instances using the AWS CLI

1. Create a notification target to receive the notification that an instance is launching. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

After you create your target, make a note of its Amazon Resource Name (ARN). For example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`.

2. Create an IAM role using the steps in [Creating a Role for an AWS Service \(AWS Management Console\)](#) in the *Using IAM* guide. When prompted to select a role type, choose **AWS Service Roles** and then select **AutoScaling Notification Access**.

After you create your role, make a note of its ARN. For example, `arn:aws:iam::123456789012:role/my-auto-scaling-role`.

3. Create a lifecycle hook perform an action (in this case, installing additional software) on the instance before it enters service. Create a lifecycle hook using the following `put-lifecycle-hook` command:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name ReadyForSoftwareIn  
stall --auto-scaling-group-name my-asg --lifecycle-transition autoscal  
ing:EC2_INSTANCE_LAUNCHING --notification-target-arn sns-topic-arn --role-  
arn iam-role-arn
```

4. When a scale out event occurs, Auto Scaling launches an instance. If the Auto Scaling group uses a configuration script, it is applied. When these steps are complete, the lifecycle hook puts the instance into a `Pending:Wait` state and uses your notification target to inform you that the instance is ready. You can connect to the instance and install additional software. For more information about connecting to an EC2 instance, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.
5. (Optional) To keep the instances in the `Pending:Wait` state until you complete the software installation, use the following `record-lifecycle-action-heartbeat` command to reset the timeout period for the instance. Note that the lifecycle action token is included in the message sent to your notification target.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-action-token  
163445fc-e180-48a4-abcc-dddec19cc2a4 --lifecycle-hook-name ReadyForSoftware  
Install --auto-scaling-group-name my-asg
```

6. When you finish installing the additional software, use the following [complete-lifecycle-action](#) command to let Auto Scaling know it can add the instance to the Auto Scaling group:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-token 163445fc-  
e180-48a4-abcc-dddec19cc2a4 --lifecycle-hook-name ReadyForSoftwareInstall  
--auto-scaling-group-name my-asg --lifecycle-action-result CONTINUE
```

## Filling a Cache of Servers

You can use Auto Scaling to fill a cache of servers ahead of an expected increase in demand. For example, you might have a schedule-based scaling policy to coincide with an upcoming marketing effort, or you might have an application that has a monthly spike in traffic. In these types of cases, it can be helpful to have EC2 instances ready in advance, because you can maximize application responsiveness, which in turn provides a better customer experience.

To cache servers, add a lifecycle hook to your Auto Scaling group that notifies you when the Auto Scaling group launches an instance. This hook applies to any new instances launched for the Auto Scaling group and keeps them in the `Pending:Wait` state.

By default, the instance remains in the `Pending:Wait` state for one hour. If you take no action during that time, Auto Scaling assumes that the instance was not configured correctly and terminates it. If you need more time, you can restart the timeout period. For example, if after 30 minutes you are not ready to add the instances to the Auto Scaling group, you can restart the timeout period, giving you a total of 90 minutes to add the instances to the Auto Scaling group. If you are ready to add the instances to the Auto Scaling group before the timeout period ends, you can complete the lifecycle action.

### Contents

- [Filling a Cache Using the AWS CLI](#) (p. 69)

## Filling a Cache Using the AWS CLI

The following steps demonstrate how to create a cache of servers for your Auto Scaling group.

### To fill a cache of servers using the AWS CLI

1. Create a notification target to receive the notification that an instance is launching. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

After you create your target, make a note of its Amazon Resource Name (ARN). For example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`.

2. Create an IAM role using the steps in [Creating a Role for an AWS Service \(AWS Management Console\)](#) in the *Using IAM* guide. When prompted to select a role type, choose **AWS Service Roles** and then select **AutoScaling Notification Access**.

After you create your role, make a note of its ARN. For example, `arn:aws:iam::123456789012:role/my-auto-scaling-role`.

3. Create a lifecycle hook to perform an action (in this case, waiting until a specific period of time elapses) on the instance before it enters service. Create a lifecycle hook using the following [put-lifecycle-hook](#) command:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name CachedServers -  
--auto-scaling-group-name my-asg --lifecycle-transition autoscaling:EC2_IN  
STANCE_LAUNCHING --notification-target-arn sns-topic-arn --role-arn iam-  
role-arn
```

4. Increase the size of your Auto Scaling group using the following [update-auto-scaling-group](#) command. This command increases the maximum size of the group to 10 and the desired capacity to 8. The new instances remain in the `Pending:Wait` state until you are ready to put them into service.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg  
--max-size 10 --desired-capacity 8
```

5. (Optional) To keep the instances in the `Pending:Wait` state until you are ready to put them into service, use the following [record-lifecycle-action-heartbeat](#) command. Note that the lifecycle action token is included in the message sent to your notification target.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-action-token  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635 --auto-scaling-group-name my-asg -  
--lifecycle-hook-name CachedServers
```

6. When you are ready to put the instances into service, use the following [complete-lifecycle-action](#) command to let Auto Scaling know that it can add the instances to the Auto Scaling group:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-token bcd2f1b8-  
9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-hook-name CachedServers --auto-  
scaling-group-name my-asg --lifecycle-action-result CONTINUE
```

## Analyzing an Instance Before Termination

A primary benefit of Auto Scaling is the ability to scale the instances for your application dynamically on an as-needed basis. As a result, instances frequently are launched and terminated without any need for manual intervention. However, you might want to understand why an instance is being terminated so that you can better configure your application's architecture. You can accomplish this by adding a lifecycle hook to your Auto Scaling group. This hook puts the instance into a `Terminating:Wait` state, while you connect to the instance and investigate the cause of the termination. The instance remains in this state until its state is set to `Terminating:Proceed`.

By default, the instance remains in the `Terminating:Wait` state for one hour. If you take no action during that time, Auto Scaling continues the termination process. If you need more time, you can restart the timeout period. For example, if after 30 minutes you discover that you need more time to analyze the instance, you can restart the timeout period, giving you a total of 90 minutes to complete your analysis. If you are ready for the instance to terminate before the timeout period ends, you can complete the lifecycle action, which continues the termination process.

When an instance has a terminating status (either `Terminating`, `Terminating:Wait`, or `Terminating:Proceed`), it is not eligible to be put back into service. If you need to troubleshoot an instance and then put it back into service, put it in a standby state before Auto Scaling initiates the termination. For more information, see [Troubleshooting Instances in an Auto Scaling Group](#) (p. 103).

## Analyzing Instances Using the AWS CLI

The following steps demonstrate the general process for putting instances in a `Terminating:Wait` state, connecting to the instance, and analyzing what might have caused the instance to fail.

### To analyze an instance before it terminates using the AWS CLI

1. Create a notification target to receive the notification that an instance is terminating. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

After you create your target, make a note of its Amazon Resource Name (ARN). For example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`.

2. Create an IAM role using the steps in [Creating a Role for an AWS Service \(AWS Management Console\)](#) in the *Using IAM* guide. When prompted to select a role type, choose **AWS Service Roles** and then select **AutoScaling Notification Access**.

After you create your role, make a note of its ARN. For example, `arn:aws:iam::123456789012:role/my-auto-scaling-role`.

3. Create a lifecycle hook to perform an action (in this case, analyze the instance) on the instance before it terminates. Create the lifecycle hook using the following [put-lifecycle-hook](#) command:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name WaitForDiagnostics
--auto-scaling-group-name my-asg --lifecycle-transition autoscaling:EC2_IN
STANCE_TERMINATING --notification-target-arn sns-topic-arn --role-arn iam-
role-arn
```

4. When an instance in the Auto Scaling group is terminated, Auto Scaling puts the instance in a `Terminating:Wait` state and sends a message to the notification target that you created. After you receive this notification, you can connect to the instance to run any diagnostics or analysis that you need. For more information about connecting to an EC2 instance, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.
5. (Optional) To keep the instances in the `Terminating:Wait` state until you complete your analysis, use the following [record-lifecycle-action-heartbeat](#) command to reset the timeout period for the instance. Note that the lifecycle action token is included in the message sent to your notification target.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-action-token
163445fc-e180-48a4-abcc-dddec19cc2a4 --auto-scaling-group-name my-asg -
-lifecycle-hook-name WaitForDiagnostics
```

6. (Optional) If you are ready for the instance to terminate before the timeout period ends, you can use the following [complete-lifecycle-action](#) command to continue the termination process:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-token 163445fc-
e180-48a4-abcc-dddec19cc2a4 --lifecycle-hook-name WaitForDiagnostics --auto-
scaling-group-name my-asg -lifecycle-action-result CONTINUE
```

## Retrieving Logs from Terminating Instances

Typically, when a scale in event occurs and Auto Scaling determines that an instance is no longer necessary, it immediately puts the instance into the `Terminating` state and you can no longer connect to the instance. The instance remains in this state until it fully terminates.

If you might want to access an instance before it is terminated and retrieve log files, you can add a lifecycle hook to your Auto Scaling group. This hook puts the instance into a `Terminating:Wait` state while you connect to the instance and retrieve the log files.

By default, the instance remains in the `Terminating:Wait` state for one hour. If you take no action during that time, Auto Scaling continues the termination process. If you need more time, you can restart the timeout period. For example, if after 30 minutes you discover that you need more time to analyze the instance, you can restart the timeout period, giving you a total of 90 minutes to retrieve the log files. If you are ready for the instance to terminate before the timeout period ends, you can complete the lifecycle action, which continues the termination process.

When an instance has a terminating status (either `Terminating`, `Terminating:Wait`, or `Terminating:Proceed`), it is not eligible to be put back into service. If you need to troubleshoot an instance and then put it back into service, put it in a standby state before Auto Scaling initiates the termination. For more information, see [Troubleshooting Instances in an Auto Scaling Group \(p. 103\)](#).

## Retrieving Logs Using the AWS CLI

The following steps demonstrate the general process to put instances in a `Terminating:Wait` state so that you can connect to the instance and download log files.

### To retrieve logs from a terminating server using the AWS CLI

1. Create a notification target to receive the notification that an instance is terminating. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

After you create your target, make a note of its Amazon Resource Name (ARN). For example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`.

2. Create an IAM role using the steps in [Creating a Role for an AWS Service \(AWS Management Console\)](#) in the *Using IAM* guide. When prompted to select a role type, choose **AWS Service Roles** and then select **AutoScaling Notification Access**.

After you create your role, make a note of its ARN. For example, `arn:aws:iam::123456789012:role/my-auto-scaling-role`.

3. Create a lifecycle hook to perform an action (in this case, retrieve log files) on the instance before it terminates. Create a lifecycle hook using the following [put-lifecycle-hook](#) command:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name GetLogs --auto-scaling-group-name my-asg --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING --notification-target-arn sns-topic-arn --role-arn iam-role-arn
```

4. When an instance in the Auto Scaling group is terminated, Auto Scaling puts the instance in a `Terminating:Wait` state and sends a message to the notification target that you created. After you receive this notification, you can connect to the instance to download the logs files that you need. For more information on how to connect to an EC2 instance, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.
5. (Optional) To keep the instance in a terminating state until you have all the data you need, use the following [record-lifecycle-action-heartbeat](#) command to reset the timeout period for the instance. Note that the lifecycle action token is included in the message sent to your notification target.

```
aws autoscaling record-lifecycle-action-heartbeat --lifecycle-action-token  
bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-hook-name GetLogs --auto-  
scaling-group-name my-asg
```

6. (Optional) If you are ready for the instance to terminate before the timeout period ends, you can use the following [complete-lifecycle-action](#) command to continue the termination process:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-token bcd2f1b8-  
9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-hook-name GetLogs --auto-scaling-  
group-name my-asg -lifecycle-action-result CONTINUE
```

## Tagging Auto Scaling Groups and Instances

You can organize and manage your Auto Scaling groups by assigning your own metadata to each group in the form of *tags*. You specify a *key* and a *value* for each tag. A key can be a general category, such as "project", "owner", or "environment", with specific associated values. For example, if one of your projects is named LIMA, you could assign a tag with a key of "project" and a value of "lima" to all Auto Scaling groups that are part of the LIMA project. Similarly, if you want to differentiate between your development environments, you could assign tags with a key of "environment" and a value of "test" to the Auto Scaling groups that are used in your test environment and assign tags with a key of "environment" and a value of "production" to Auto Scaling groups that are used in your production environment. We recommend that you use a consistent set of tags to make it easier to track your Auto Scaling groups.

You can also specify that the tags for your Auto Scaling groups are added to the EC2 instances launched in the group. Tagging your EC2 instances enables you to see instance cost allocation by tag in your AWS bill. For example, you can track the cost of running Auto Scaling instances for project LIMA in a test environment. For more information, see [Use Cost Allocation Tags](#) in the *AWS Billing and Cost Management User Guide*.

### Contents

- [Tag Restrictions](#) (p. 73)
- [Add or Modify Tags for Your Auto Scaling Group](#) (p. 74)
- [Delete Tags](#) (p. 75)

## Tag Restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 10.
- The maximum key length is 127 Unicode characters.
- The maximum value length is 255 Unicode characters.
- Tag keys and values are case sensitive.
- 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 against toward your limit of tags per Auto Scaling group.

Note that when you launch an instance in an Auto Scaling group, Auto Scaling adds a tag to the instance with a key of `aws:autoscaling:groupName` and a value of the name of the Auto Scaling group.

You can create and assign tags to your Auto Scaling group when you either create or update your Auto Scaling group. You can remove Auto Scaling group tags at any time. For information about assigning tags when you create your Auto Scaling group, see [Step 2: Create an Auto Scaling Group \(p. 17\)](#).

## Add or Modify Tags for Your Auto Scaling Group

When you add a tag to your Auto Scaling group, you can specify whether it should be added to instances launched in your 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 Using the AWS Management Console \(p. 74\)](#)
- [Add or Modify Tags Using the AWS CLI \(p. 74\)](#)

## Add or Modify Tags Using the AWS Management Console

Use the Amazon EC2 console to add or modify tags.

### To add or modify tags

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. In the bottom pane, select the **Tags** tab.
5. Click **Add/Edit tags**. The **Add/Edit Auto Scaling Group Tags** dialog box lists any existing tags for the Auto Scaling group.
6. To modify existing tags, edit the **Key** and **Value** fields.
7. To add a new tag, click **Add tag** and edit the **Key** and **Value** fields. 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.
8. When you have finished adding tags, click **Save**.

## Add or Modify Tags Using the AWS CLI

Use the [create-or-update-tags](#) command to create or modify a tag. For example, the following command adds a tag with a key of "environment" and a value of "test" that will also be added to instances launched in the Auto Scaling group after this change. If a tag with this key already exists, the existing tag is replaced.

```
aws autoscaling create-or-update-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-group,Key=environment,Value=test,PropagateAtLaunch=true"
```

The following is an example response:

```
OK-Created/Updated tags
```

Use the following [describe-tags](#) command to list the tags for the specified Auto Scaling 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": "test",
      "Key": "environment"
    }
  ]
}
```

Alternatively, use the following [describe-auto-scaling-groups](#) command to verify that the tag is added to the Auto Scaling group.

```
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": "test",
          "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 Using the AWS Management Console \(p. 76\)](#)
- [Delete Tags Using the AWS CLI \(p. 76\)](#)



## Delete Tags Using the AWS Management Console

To delete a tag using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. In the bottom pane, select the **Tags** tab.
5. Click **Add/Edit tags**. The **Add/Edit Auto Scaling Group Tags** dialog box lists any existing tags for the Auto Scaling group.
6. Click the delete icon next to the tag.
7. Click **Save**.

## Delete Tags Using the 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"
```

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

## Launching Spot Instances in Your Auto Scaling Group

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. You can set up Auto Scaling to launch Spot Instances instead of On-Demand instances.

Before launching Spot Instances using Auto Scaling, we recommend that you become familiar with launching and managing Spot Instances using Amazon EC2. For more information, see [Spot Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Here's how Spot Instances work with Auto Scaling:

- **Setting your bid price.** When you use Auto Scaling to launch Spot Instances, you set your bid price in the launch configuration. You can't use a single launch configuration to launch both On-Demand instances and Spot Instances.
- **Changing your bid price.** To change your Spot bid price, you must create a new launch configuration with the new bid price, and then associate it with your Auto Scaling group. Note that the existing instances continue to run as long as the bid price specified in the launch configuration used for those instances is higher than the current Spot market price.
- **Spot market price and your bid price.** If the market price for Spot Instances rises above your Spot bid price for a running instance in your Auto Scaling group, Amazon EC2 terminates your instance. If your Spot bid price exactly matches the Spot market price, whether your bid is fulfilled depends on several factors—such as available Spot Instance capacity.
- **Maintaining your Spot Instances.** When your Spot Instance is terminated, Auto Scaling attempts to launch a replacement instance to maintain the desired capacity for the group. If the bid price is higher

than the market price, then a Spot Instance is launched. Otherwise, no instance is launched, but Auto Scaling keeps trying.

- **Auto Scaling and Spot Instance termination.** Auto Scaling can terminate or replaces Spot Instances just as it can terminate or replace On-Demand instances. For more information, see [Choosing a Termination Policy for Your Auto Scaling Group](#) (p. 26).

### Contents

- [Launching Spot Instances Using the AWS Management Console](#) (p. 77)
- [Launching Spot Instances Using the AWS CLI](#) (p. 80)

## Launching Spot Instances Using the AWS Management Console

To create an Auto Scaling group that launches Spot Instances, complete the following tasks:

### Tasks

- [Create a Launch Configuration](#) (p. 77)
- [Create an Auto Scaling Group](#) (p. 78)
- [Verify and Check Your Instances](#) (p. 78)
- [\(Optional\) Get Notifications When the Auto Scaling Group Changes](#) (p. 79)
- [\(Optional\) Update the Bid Price](#) (p. 79)
- [Clean Up](#) (p. 80)

## Create a Launch Configuration

### To create a launch configuration

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under Auto Scaling, click **Launch Configurations**. If you are new to Auto Scaling, you see a welcome page; click **Create Auto Scaling group**.
3. Click **Create launch configuration**.
4. On the **Choose AMI** page, select an AMI.
5. On the **Choose Instance Type** page, select a hardware configuration for your example. Click **Next: Configure details**.

#### Note

T2 instances must be launched into a subnet of a VPC. If you select a `t2.micro` instance but don't have a VPC, one is created for you. This VPC includes a public subnet in each Availability Zone in the region.

6. On the **Configure Details** page, do the following:
  - a. In the **Name** field, enter a name for your launch configuration. Consider including "Spot" and the bid price in this name.
  - b. Select **Request Spot Instances**. When you select this option, you'll see the current prices for the Availability Zones in the region. Enter your bid price in **Maximum price**.

## Auto Scaling Developer Guide

### Launching Spot Instances Using the AWS Management Console

Purchasing option ☒ Request Spot Instances

Current price

us-east-1a	0.007
us-east-1b	0.007
us-east-1c	0.007
us-east-1d	0.007
us-east-1e	0.007

Maximum price \$ 0.05

- Under **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to you instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.
  - Click **Skip to review**.
- On the **Review** page, click **Edit security groups**, follow the instructions to choose an existing security group, and then click **Review**.
  - On the **Review** page, notice that the launch configuration details include your bid price. Click **Create launch configuration**.

Create Launch Configuration

AMI Details

Amazon Linux AMI 2015.09.1 - ami-83e4bcea

Instance Type

Instance Type	ECUs	VCPU	Memory GB	Instance Storage (GiB)	EBS-Optimized	Network Performance
m1.small	1	1	1.7	1 x 160	-	Low

Launch configuration details

Name: spot-Scents

Purchasing option: Spot Request

Maximum price: \$0.05

EBS Optimized: No

Monitoring: No

RAM role: None

Tenancy: Shared tenancy (multi-tenant hardware)

Kernel ID: Use default

RAM Disk ID: Use default

Buttons: Cancel, Previous, Create launch configuration

- In the **Select an existing key pair or create a new key pair** dialog box, select one of the listed options. Click the acknowledgment check box, and then click **Create launch configuration**.

#### Warning

Do not select **Proceed without a key pair** if you need to connect to your instance.

## Create an Auto Scaling Group

When you create your Auto Scaling group, you must specify the launch configuration that you just created. Remember that the launch configuration is a template for your instances, and it includes the bid price for your Spot Instances.

For step-by-step directions for creating your Auto Scaling group using the AWS Management Console, see [Create an Auto Scaling Group \(p. 55\)](#).

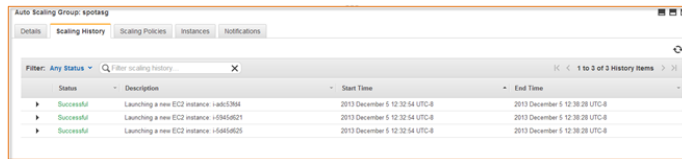
## Verify and Check Your Instances

### To confirm that Auto Scaling is launching your Spot Instances

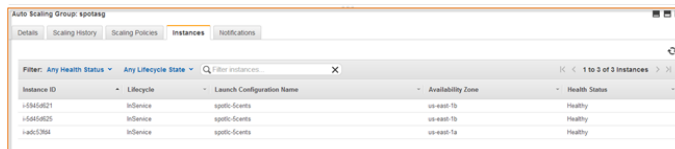
- Select your new Auto Scaling group.
- Click the **Scaling History** tab. It shows that Auto Scaling successfully launched the Spot Instances that you requested.

## Auto Scaling Developer Guide

### Launching Spot Instances Using the AWS Management Console



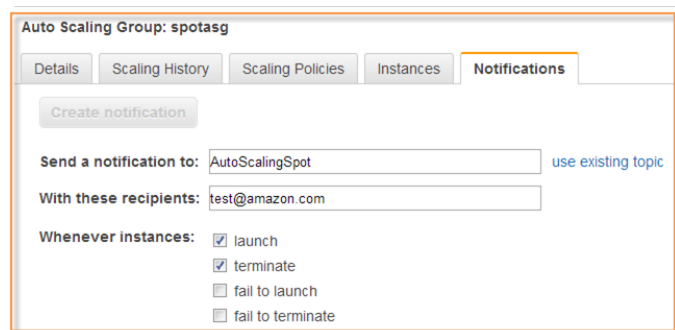
- To get details about your Spot Instances, click the **Instances** tab. You'll see that Auto Scaling is launching the instances you requested in the Availability Zones that you specified.



## (Optional) Get Notifications When the Auto Scaling Group Changes

### To set up notifications

- Select the Auto Scaling group, and then select the **Notifications** tab.
- Click **Create notification**.
- Click **create topic**, specify the following, and then click **Save**:
  - Send a notification to** - AutoScalingSpot
  - With these recipients** - *your email account*
  - Whenever instances** - launch, terminate



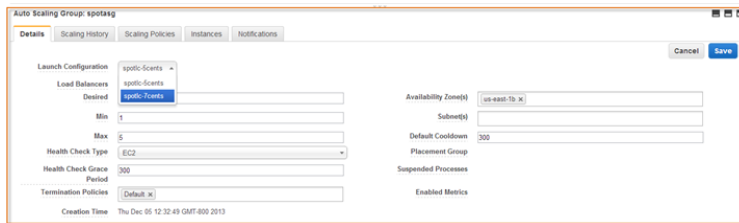
As soon as your notification topic is created, the email account you specified receives an email confirmation.

## (Optional) Update the Bid Price

### To update the bid price for the Spot Instances

- Create a launch configuration with the same specifications as in [Create a Launch Configuration \(p. 77\)](#), but with a different name and maximum price.
- Select your Auto Scaling group.
- On the **Details** tab, click **Edit**.

4. Select the new launch configuration that you just created, and then click **Save**.



## Clean Up

After you're finished using your instances and your Auto Scaling group, it is a good practice to clean up. When you delete an Auto Scaling group, this also deletes all the Spot Instances and outstanding Spot bids for the group.

### To clean up Auto Scaling group and instances

1. Select your Auto Scaling group.
2. Click **Actions**, and then click **Delete**.
3. When prompted for confirmation, click **Yes, Delete**.

## Launching Spot Instances Using the AWS CLI

To create an Auto Scaling group that launches Spot Instances, complete the following tasks:

### Tasks

- [Create a Launch Configuration \(p. 80\)](#)
- [Create an Auto Scaling Group \(p. 80\)](#)
- [Verify and Check Your Instances \(p. 81\)](#)
- [\(Optional\) Get Notifications When the Auto Scaling Group Changes \(p. 83\)](#)
- [\(Optional\) Update the Bid Price for the Spot Instances \(p. 83\)](#)
- [Clean Up \(p. 84\)](#)

## Create a Launch Configuration

To place bids for Spot Instances using Auto Scaling, specify the maximum price you are willing to pay for an instance by using the `--spot-price` option with the `create-launch-configuration` command as follows:

```
aws autoscaling create-launch-configuration --launch-configuration-name spot-1c-5cents --image-id ami-1a2bc4d --instance-type m1.small --spot-price "0.05"
```

## Create an Auto Scaling Group

Create your Auto Scaling group using the `create-auto-scaling-group` command with the launch configuration that you just created. The following command launches 2 Spot Instances:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name spot-asg -  
-launch-configuration-name spot-lc-5cents --availability-zones "us-west-2a"  
"us-west-2b" --max-size 5 --min-size 1 --desired-capacity 2
```

## Verify and Check Your Instances

Use the [describe-scaling-activities](#) command as follows to list the activities that Auto Scaling performed for your Auto Scaling group:

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name spot-asg
```

If both bids can't be fulfilled initially, the output looks similar to the following example, where one bid is successful and Auto Scaling is waiting for the other bid:

```
{  
  "Activities": [  
    {  
      "Description": "Placing Spot instance request. Status Reason: Placed  
Spot instance request: sir-036wjsp9. Waiting for instance(s)",  
      "AutoScalingGroupName": "spot-asg",  
      "ActivityId": "28189e6b-e14f-4783-8d48-4d03b40b1354",  
      "Details": "{\"Availability Zone\":\"us-west-2a\"}",  
      "StartTime": "2015-03-01T16:21:41.578Z",  
      "Progress": 20,  
      "Cause": "At 2015-03-01T16:21:40Z a difference between desired and  
actual capacity changing the desired capacity, increasing the capacity from 0  
to 2.",  
      "StatusMessage": "Placed Spot instance request: sir-036wjsp9.  
Waiting for instance(s)",  
      "StatusCode": "WaitingForSpotInstanceId"  
    },  
    {  
      "Description": "Launching a new EC2 instance: i-d95eb0d4",  
      "AutoScalingGroupName": "spot-asg",  
      "ActivityId": "b987ab02-f7c3-4948-a0bc-5d1449de30ec",  
      "Details": "{\"Availability Zone\":\"us-west-2b\"}",  
      "StartTime": "2015-03-01T16:21:41.578Z",  
      "Progress": 100,  
      "EndTime": "2015-03-01T16:29:46Z",  
      "Cause": "At 2015-03-01T16:21:40Z a difference between desired and  
actual capacity changing the desired capacity, increasing the capacity from 0  
to 2.",  
      "StatusCode": "Successful"  
    }  
  ]  
}
```

If the output of `as-describe-scaling-activities` includes `Failed` activities, check the response for details. For example, it's possible that the AMI ID is no longer valid or that it's incompatible with the instance type that you selected. If no reason is given, check whether your bid price is above the Spot market price for that Availability Zone.

To view information about the instances for your Auto Scaling group, use the [describe-auto-scaling-groups](#) command as follows:

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

The following is example output that shows the Auto Scaling launched two instances, as you specified, and they are both running:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 0,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "spot-asg",
      "DefaultCooldown": 300,
      "MinSize": 1,
      "Instances": [
        {
          "InstanceId": "i-d95eb0d4",
          "AvailabilityZone": "us-west-2b",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "spot-lc-5cents"
        },
        {
          "InstanceId": "i-13d7dc1f",
          "AvailabilityZone": "us-west-2a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "spot-lc-5cents"
        }
      ],
      "MaxSize": 5,
      "VPCZoneIdentifier": null,
      "TerminationPolicies": [
        "Default"
      ],
      "LaunchConfigurationName": "spot-lc-5cents",
      "CreatedTime": "2015-03-01T16:12:35.608Z",
      "AvailabilityZones": [
        "us-west-2b",
        "us-west-2a"
      ],
      "HealthCheckType": "EC2"
    }
  ]
}
```

In addition to using `describe-auto-scaling-groups`, you can use the [describe-auto-scaling-instances](#) command as follows:

```
aws autoscaling describe-auto-scaling-instances
```

The following is example output:

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-west-2a",
      "InstanceId": "i-13d7dc1f",
      "AutoScalingGroupName": "spot-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "spot-lc-5cents"
    },
    {
      "AvailabilityZone": "us-west-2b",
      "InstanceId": "i-d95eb0d4",
      "AutoScalingGroupName": "spot-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "spot-lc-5cents"
    }
  ]
}
```

## (Optional) Get Notifications When the Auto Scaling Group Changes

For information about setting up email notifications in Auto Scaling, see [Getting Notifications When Your Auto Scaling Group Changes](#) (p. 119).

## (Optional) Update the Bid Price for the Spot Instances

### To update the bid price for Spot Instances

1. Create a launch configuration with the same specifications as before, but with a different name and maximum price, as follows:

```
aws autoscaling create-launch-configuration --launch-configuration-name
spot-lc-7cents --image-id ami-1a2b3c4d --instance-type m1.small --spot-price
"0.07"
```

2. Modify your Auto Scaling group to use the new launch configuration, by using the [update-auto-scaling-group](#) command as follows:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name spot-
asg --launch-configuration-name spot-lc-7cents
```

3. View your changes using the [describe-scaling-activities](#) command as follows:

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name spot-
asg
```



## Clean Up

After you're finished using your instances and your Auto Scaling group, it is a good practice to clean up. Use the `delete-auto-scaling-group` command as follows with the optional `--force-delete` parameter, which specifies that EC2 instances that are part of the Auto Scaling group are terminated with the Auto Scaling group, even if the instances are still running. Otherwise, you must terminate these instances before you can delete your Auto Scaling group.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name spot-asg --force-delete
```

# Configuring Your Auto Scaling Groups

---

After you create an Auto Scaling group within your network architecture, you may find that there are other actions you might want to take. For example, you might want to:

## [Load balance your Auto Scaling group \(p. 86\)](#)

A load balancer is an important part of Auto Scaling, as it allows you to distribute traffic across the instances within the Auto Scaling group. Auto Scaling works particularly well with Elastic Load Balancing; however, you can also use Auto Scaling with the load balancer of your choice.

## [Attach an existing instance to your Auto Scaling group \(p. 95\)](#)

As you continue to refine and improve your application, you might want to launch and configure an EC2 instance and then attach it to your Auto Scaling group. This is particularly useful if you want to test certain changes before you update all of the instances in the Auto Scaling group.

## [Detach an instances from an Auto Scaling group \(p. 98\)](#)

Occasionally, you might find that you want to move instances out of an Auto Scaling group. This could be because you want to move the instances into a different Auto Scaling group, or because you no longer want to use Auto Scaling in that particular area of your application.

## [Merge Auto Scaling groups from different Availability Zones \(p. 101\)](#)

It is not uncommon to start with a couple of Auto Scaling groups, each residing in a single Availability Zone. However, a more efficient implementation would have a single Auto Scaling group that spans multiple Availability Zones. This involves modifying an existing Auto Scaling group and then terminating the obsolete groups.

## [Temporarily remove instances from an Auto Scaling group \(p. 103\)](#)

Sometimes, you might want to move an instance from your application, but still have it managed by the Auto Scaling group. For example, you might want to install a patch to existing instances in your Auto Scaling group, and don't want to relaunch the instances. You might want to update only a few your instances, so you can see in real time which configuration settings work best. Auto Scaling supports temporarily removing instances from receiving traffic, and then putting them back in service when you're ready.

## [Suspend and resume your Auto Scaling group \(p. 110\)](#)

Auto Scaling allows you to retain complete control over your network architecture. If you discover that you need to investigate a configuration or other issue, you can suspend Auto Scaling actions and then resume them again when your investigation concludes.

## [Shut down an Auto Scaling group \(p. 112\)](#)

You can choose to shut down an Auto Scaling group at any time.

If you haven't yet created an Auto Scaling group, you might want to review the following sections:

- [What Is Auto Scaling? \(p. 1\)](#)—Describes the core concepts that you should understand before adding Auto Scaling to your network infrastructure.
- [Getting Started with Auto Scaling \(p. 16\)](#)—Create an Auto Scaling group and see how it can help your applications become more highly available and fault tolerant.
- [Planning Your Auto Scaling Group \(p. 21\)](#)—Describes how to create launch configurations, build Auto Scaling groups, and perform other tasks associated with creating Auto Scaling groups.

## Load Balance Your Auto Scaling Group

When you use Auto Scaling, you can automatically increase the number of EC2 instances you're using when the user demand goes up, and you can decrease the number of EC2 instances when demand goes down. As Auto Scaling dynamically adds and removes EC2 instances, you need to ensure that the traffic coming to your web application is distributed across all of your running EC2 instances. AWS provides the Elastic Load Balancing service to distribute the incoming web traffic (called the *load*) automatically among all the EC2 instances that you are running. Elastic Load Balancing manages incoming requests by optimally routing traffic so that no one instance is overwhelmed. Using Elastic Load Balancing with your auto-scaled web application makes it easy to route traffic among your dynamically changing fleet of EC2 instances.

You can use Elastic Load Balancing to route traffic to EC2 instances in your Auto Scaling group. For more information about Elastic Load Balancing, see [What Is Elastic Load Balancing?](#) in the *Elastic Load Balancing Developer Guide*.

Elastic Load Balancing uses load balancers to monitor traffic and handle requests that come through the Internet. To use Elastic Load Balancing with your Auto Scaling group, you first create a load balancer and then register your Auto Scaling group with the load balancer. Your load balancer acts as a single point of contact for all incoming traffic. You can register multiple load balancers with a single Auto Scaling group. For information about registering your load balancer with your Auto Scaling group, see [Set Up a Scaled and Load-Balanced Application \(p. 87\)](#).

Elastic Load Balancing sends data about your load balancers and EC2 instances to Amazon CloudWatch. CloudWatch collects the data and presents it as readable, near-time metrics. After registering the load balancer with your Auto Scaling group, you can configure your Auto Scaling group to use Elastic Load Balancing metrics (such as request latency or request count) to scale your application automatically. For information about Elastic Load Balancing metrics, see [Monitor Your Load Balancer Using Amazon CloudWatch](#). For information about using CloudWatch metrics to scale automatically, see [Dynamic Scaling \(p. 32\)](#).

By default, the Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks. Elastic Load Balancing also performs health checks on the EC2 instances that are registered with the load balancer. After you've registered your Auto Scaling group with a load balancer, you can choose to use the results of the Elastic Load Balancing health check in addition to the EC2 instance status checks to determine the health of the EC2 instances in your Auto Scaling group. For information about adding an Elastic Load Balancing health check, see [Add an Elastic Load Balancing Health Check to your Auto Scaling Group \(p. 91\)](#).

If connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete or for the maximum timeout to expire, whichever comes first, before terminating instances due to a scaling event or health check replacement. For information about connection draining, see [Connection Draining](#) in the *Elastic Load Balancing Developer Guide*.

You can take advantage of the safety and reliability of geographic redundancy by spanning your Auto Scaling groups across multiple Availability Zones within a region and then setting up load balancers to distribute incoming traffic across those Availability Zones. For information about expanding your auto-scaled

and load-balanced application to an additional Availability Zone, see [Expand Your Scaled and Load-Balanced Application to an Additional Availability Zone](#) (p. 92).

## Set Up a Scaled and Load-Balanced Application

You can register your Auto Scaling group with a load balancer to set up an auto-scaled and load-balanced application.

### Prerequisites

Before you begin, create a load balancer. You don't need to register your EC2 instances with your load balancer, as Auto Scaling launches the instances and then attaches the group to the load balancer. For more information about creating a load balancer, see [Get Started With Elastic Load Balancing](#) in the *Elastic Load Balancing Developer Guide*.

### Contents

- [Setting Up the Application Using the AWS Management Console](#) (p. 87)
- [Setting Up an Application Using the AWS CLI](#) (p. 89)

## Setting Up the Application Using the AWS Management Console

Complete the following tasks to set up a scaled and load-balanced application.

### Tasks

- [Create or Select a Launch Configuration](#) (p. 87)
- [Create an Auto Scaling Group](#) (p. 88)
- [\(Optional\) Verify that Your Auto Scaling Group Launched with Your Load Balancer](#) (p. 89)

### Create or Select a Launch Configuration

If you already have a launch configuration that you'd like to use, select it using the following procedure.

#### To select an existing launch configuration

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar at the top of the screen, the current region is displayed. Select the region that you used to create your load balancer.
3. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**.
4. Select the launch configuration.
5. Click **Create Auto Scaling group**.

Alternatively, to create a new launch configuration, use the following procedure.

#### To create a launch configuration

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar at the top of the screen, the current region is displayed. Select the region that you used to create your load balancer.
3. In the navigation pane, under **Auto Scaling**, click **Launch Configurations**. If you don't have any Auto Scaling resources, you see a welcome page; click **Create Auto Scaling group**.
4. Click **Create launch configuration**.

5. On the **Choose AMI** page, select an AMI.
6. On the **Choose Instance Type** page, select a hardware configuration for your instance. Click **Next: Configure details**.

**Note**

T2 instances must be launched into a subnet of a VPC. If you select a `t2.micro` instance but don't have a VPC, one is created for you. This VPC includes a public subnet in each Availability Zone in the region.

7. On the **Configure Details** page, do the following:
  - a. In the **Name** field, enter a name for your launch configuration.
  - b. If you need to connect to an instance in a nondefault VPC, you must expand **Advanced Details** and then select **Assign a public IP address to every instance**.
  - c. (Optional) To specify user data or a configuration script for your instance, expand **Advanced Details** and copy the data or script in the **User data** field.
  - d. Click **Skip to review**.
8. On the **Review** page, click **Edit security groups**, follow the instructions to choose an existing security group, and then click **Review**.
9. On the **Review** page, click **Create launch configuration**.
10. In the **Select an existing key pair or create a new key pair** dialog box, select one of the listed options. Click the acknowledgment check box, and then click **Create launch configuration**.

**Warning**

Do not select **Proceed without a key pair** if you need to connect to your instance.

11. The **Launch configuration creation status** page displays the status of your newly created launch configuration. Click **Create an Auto Scaling group using this launch configuration**.

## Create an Auto Scaling Group

Use the following procedure to continue where you left off after selecting or creating your launch configuration.

### To create an Auto Scaling group

1. On the **Configure Auto Scaling group details** page, do the following:
  - a. In **Group name**, enter a name for your Auto Scaling group.
  - b. In **Group size**, enter the number of instances for your group to start with.
  - c. If you are launching a T2 instance, you must select a VPC in **Network**. Otherwise, if your account supports EC2-Classic and you are launching a type of instance that doesn't require a VPC, you can select either `Launch into EC2-Classic` or a VPC.
  - d. If you selected a VPC in the previous step, select a subnet from **Subnet**. If you selected EC2-Classic in the previous step, select an Availability Zone from **Availability Zone(s)**.
  - e. Under **Advanced Details**, select `Receive traffic from Elastic Load Balancer(s)`. When you select this option, you'll see an additional empty field. Click the field and select your load balancer.
  - f. (Optional) To use Elastic Load Balancing health checks, select **ELB**.

▼ Advanced Details

**Load Balancing** ⓘ ☒ Receive traffic from Elastic Load Balancer(s)  

my-as-lb x

**Health Check Type** ⓘ ☒ ELB ☐ EC2

**Health Check Grace Period** ⓘ 

300

 seconds

**Monitoring** ⓘ Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration my-test-ic. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency.  
[Learn more](#)

[Cancel](#) [Next: Configure scaling policies](#)

g. Click **Next: Configure scaling policies**.

2. In the **Configure scaling policies** page, select **Keep this group at its initial size**, and then click **Review**.

If you want to configure scaling policies for your Auto Scaling group, see [Scaling Based on Metrics \(p. 34\)](#).

3. Review the details of your Auto Scaling group. You can click **Edit** to make changes. When you are finished, click **Create Auto Scaling group**.

## (Optional) Verify that Your Auto Scaling Group Launched with Your Load Balancer

**To verify that your Auto Scaling group has launched with your load balancer**

1. Select your Auto Scaling group.
2. In the bottom pane, click the **Details** tab. The **Load Balancers** field displays the name of your load balancer.
3. Click the **Scaling History** tab. The **Status** column shows you the instances launched by your Auto Scaling group. While an instance is launching, its status is `In progress`. The status changes to `Successful` after the instance is launched.
4. Click the **Instances** tab. The **Lifecycle** column shows you the state of your newly launched instances. After the instance starts, its state is `InService`.

The **Health Status** column shows the result of the health checks on your instances.

## Setting Up an Application Using the AWS CLI

Complete the following tasks to set up a scaled and load-balanced application.

### Tasks

- [Create a Launch Configuration \(p. 89\)](#)
- [Create an Auto Scaling Group with a Load Balancer \(p. 90\)](#)
- [\(Optional\) Verify That Your Auto Scaling Group Launched with a Load Balancer \(p. 90\)](#)

## Create a Launch Configuration

If you already have a launch configuration that you'd like to use, skip this step.

**To create the launch configuration**

Use the following [create-launch-configuration](#) command:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc
--image-id ami-514ac838 --instance-type m1.small
```

## Create an Auto Scaling Group with a Load Balancer

You can attach an existing load balancer to an Auto Scaling group when you create the group.

### To create an Auto Scaling group with a load balancer

Use the following [create-auto-scaling-group](#) command with the `--load-balancer-names` option to create a group with a load balancer:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-lb-asg
--launch-configuration-name my-lc --availability-zones
"us-west-2a" "us-west-2b" --load-balancer-names "my-lb" --max-size 5 --min-size
1 --desired-capacity 2
```

## (Optional) Verify That Your Auto Scaling Group Launched with a Load Balancer

After you have created an Auto Scaling group with a load balancer, you can verify that the load balancer has been launched with the group.

### To verify that your Auto Scaling group launched with a load balancer

Use the following [describe-auto-scaling-groups](#) command:

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

The following is example output showing a group with a load balancer and two running instances:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 0,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [
        "my-lb"
      ],
      "AutoScalingGroupName": "my-lb-asg",
      "DefaultCooldown": 300,
      "MinSize": 1,
      "Instances": [
        {
          "InstanceId": "i-d95eb0d4",
          "AvailabilityZone": "us-west-2b",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",

```

```
        "LaunchConfigurationName": "my-lc"
      },
      {
        "InstanceId": "i-13d7dc1f",
        "AvailabilityZone": "us-west-2a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      }
    ],
    "MaxSize": 5,
    "VPCZoneIdentifier": null,
    "TerminationPolicies": [
      "Default"
    ],
    "LaunchConfigurationName": "my-lc",
    "CreatedTime": "2015-03-01T16:12:35.608Z",
    "AvailabilityZones": [
      "us-west-2b",
      "us-west-2a"
    ],
    "HealthCheckType": "EC2"
  }
]
```

## Add an Elastic Load Balancing Health Check to your Auto Scaling Group

By default, an Auto Scaling group periodically reviews the results of EC2 instance status to determine the health state of each instance. However, if you have associated your Auto Scaling group with an Elastic Load Balancing load balancer, you can choose to use the Elastic Load Balancing health check. In this case, Auto Scaling determines the health status of your instances by checking the results of both the EC2 instance status check and the Elastic Load Balancing instance health check.

For information about EC2 instance status checks, see [Monitor Instances With Status Checks](#) in the *Amazon EC2 User Guide for Linux Instances*. For information about Elastic Load Balancing health checks, see [Health Check](#) in the *Elastic Load Balancing Developer Guide*.

Auto Scaling marks an instance unhealthy if calls to the Amazon EC2 action [DescribeInstanceStatus](#) return any state other than `running`, the system status shows `impaired`, or calls to Elastic Load Balancing action [DescribeInstanceHealth](#) return `OutOfService` as the instance state.

If there are multiple load balancers associated with your Auto Scaling group, Auto Scaling checks the health state of your EC2 instances by making health check calls to each load balancer. For each call, if the Elastic Load Balancing action returns any state other than `InService`, the instance is marked as unhealthy. After Auto Scaling marks an instance as unhealthy, it remains in that state, even if subsequent calls from other load balancers return an `InService` state for the same instance.

Frequently, new instances need to warm up briefly before they can pass a health check. To provide ample warm-up time, set the health check grace period of the group to cover the expected startup period of your application. Auto Scaling waits until the grace period ends before checking the health status of the instance. The grace period starts after the instance passes the EC2 system status check and instance status check. If you add a lifecycle hook to perform actions as your instances launch, the grace period must allow enough time to complete the lifecycle hook.



The following examples show you how to add an Elastic Load Balancing health check to your Auto Scaling group, assuming that you have created a load balancer and have registered the load balancer with your Auto Scaling group. If you have not registered the load balancer with your Auto Scaling group, see [Set Up a Scaled and Load-Balanced Application](#) (p. 87).

#### Contents

- [Adding a Health Check Using the Console](#) (p. 92)
- [Adding a Health Check Using the AWS CLI](#) (p. 92)

## Adding a Health Check Using the Console

When Auto Scaling checks the health status, it ignores instances that have been in the `InService` state for less than the number of seconds you specified for the health check grace period.

Use the following procedure to create a health check with a grace period of 300 seconds.

#### To add an Elastic Load Balancing health check using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your group.
4. In the bottom pane, on the **Details** tab, click **Edit**.
5. In the **Health Check Type** list, select **ELB**.
6. In the **Health Check Grace Period** field, enter **300**.
7. Click **Save**.
8. In the bottom pane, select the **Instances** tab. The **Health Status** column displays the results of the newly added Elastic Load Balancing health check. If the calls to Elastic Load Balancing health check for the instance returns any state other than `InService`, Auto Scaling marks the instance as `Unhealthy`. And if the instance is marked as `Unhealthy`, Auto Scaling starts the termination process for the instance.

## Adding a Health Check Using the AWS CLI

Use the following `update-auto-scaling-group` command to create a health check with 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
```

When Auto Scaling checks the health status, it ignores instances that have been in the `InService` state for less than the number of seconds you specified using `--health-check-grace-period`.

## Expand Your Scaled and Load-Balanced Application to an Additional Availability Zone

When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the Availability Zones for your Auto Scaling group. Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Auto Scaling attempts to launch in other Availability Zones until it succeeds.

An Auto Scaling group can contain EC2 instances that come from one or more Availability Zones within the same region. However, an Auto Scaling group cannot span multiple regions.

You can set up your load balancer to distribute incoming requests across EC2 instances in a single Availability Zone or multiple Availability Zones within a region. The load balancer does not distribute traffic across regions. For critical applications, we recommend that you distribute incoming traffic across multiple Availability Zones by registering your Auto Scaling group in multiple Availability Zones and then enabling your load balancer in each of those Availability Zones. Incoming traffic is load balanced equally across all the Availability Zones enabled for your load balancer.

If your load balancer detects unhealthy EC2 instances in an enabled Availability Zone, it stops routing traffic to those instances. Instead, it spreads the load across the remaining healthy instances. If all instances in an Availability Zone are unhealthy, but you have instances in other Availability Zones, Elastic Load Balancing routes traffic to your registered and healthy instances in those other Availability Zones. It resumes load balancing to the original instances when they have been restored to a healthy state and are registered with your load balancer.

You can expand the availability of your scaled and load-balanced application by adding a new Availability Zone to your Auto Scaling group and then enabling that Availability Zone for your load balancer. After you've enabled the new Availability Zone, the load balancer begins to route traffic equally among all the enabled Availability Zones.

## Expanding Applications Using the AWS CLI

In this example, you expand the availability of your application in EC2-Classic to an additional Availability Zone.

### To expand a scaled, load-balanced application to an additional Availability Zone

1. Update the Auto Scaling group using the following `update-auto-scaling-group` command:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-lb-  
asg --availability-zone-names "us-west-2a" "us-west-2b" "us-west-2c" --min-  
size 3
```

2. Verify that the instances in the new Availability Zone 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-  
lb-asg
```

The following is example output that indicates that the instances are ready:

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupARN": "arn",  
      "HealthCheckGracePeriod": 300,  
      "SuspendedProcesses": [],  
      "DesiredCapacity": 3,  
      "Tags": [],  
      "EnabledMetrics": [],  
      "LoadBalancerNames": [  
        "my-lb"  
      ],  
    },  
  ],  
}
```

## Auto Scaling Developer Guide

### Expand Your Scaled and Load-Balanced Application to an Additional Availability Zone

```
"AutoScalingGroupName": "my-lb-asg",
"DefaultCooldown": 300,
"MinSize": 3,
"Instances": [
  {
    "InstanceId": "i-9823ca95",
    "AvailabilityZone": "us-west-2b",
    "HealthStatus": "Healthy",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  },
  {
    "InstanceId": "i-a42d27a8",
    "AvailabilityZone": "us-west-2a",
    "HealthStatus": "Healthy",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  },
  {
    "InstanceId": "i-545dc19d",
    "AvailabilityZone": "us-west-2c",
    "HealthStatus": "Healthy",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  }
],
"MaxSize": 6,
"VPCZoneIdentifier": null,
"TerminationPolicies": [
  "Default"
],
"LaunchConfigurationName": "my-lc",
"CreatedTime": "2015-03-02T01:28:08.398Z",
"AvailabilityZones": [
  "us-west-2c",
  "us-west-2b",
  "us-west-2a"
],
"HealthCheckType": "ELB"
}
]
```

3. Update the load balancer to route traffic to the new Availability Zone using the following `enable-availability-zones-for-load-balancer` command. Traffic is routed equally among all the enabled Availability Zones.

```
aws elb enable-availability-zones-for-load-balancer --load-balancer-name
my-lb --availability-zones us-west-2c
```

The following is example output:

```
{
  "AvailabilityZones": [
    "us-west-2a",
    "us-west-2b",
```

```
    "us-west-2c"  
  ]  
}
```

## Attach EC2 Instances to Your Auto Scaling Group

Auto Scaling provides you with an option to enable Auto Scaling for one or more EC2 instances by attaching them to your existing Auto Scaling group. After the instances are attached, they become a part of the Auto Scaling group.

The instance that you want to attach must meet the following criteria:

- The instance is in the `running` state.
- The AMI used to launch the instance must still exist.
- The instance is not a member of another Auto Scaling group.
- The instance is in the same Availability Zone as the Auto Scaling group.
- If the Auto Scaling group is associated with a load balancer, the instance and the load balancer must both be in EC2-Classic or the same VPC.

When you attach instances, Auto Scaling increases the desired capacity of the group 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.

### Contents

- [Attaching an Instance Using the AWS Management Console \(p. 95\)](#)
- [Attaching an Instance Using the AWS CLI \(p. 96\)](#)

Note that 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
- Availability Zone = `us-east-1a`

## Attaching an Instance Using the AWS Management Console

You can attach an existing instance to an existing Auto Scaling group, or to a new Auto Scaling group as you create it.

### To attach an instance to a new Auto Scaling group using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, select **Instances**.

3. Select the instance, click **Actions**, select **Instance Settings**, and then click **Attach to Auto Scaling Group**.
4. In the **Attach to Auto Scaling Group** dialog box, select a **new Auto Scaling group**, enter a name for the group, and then click **Attach**.

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 launch configuration gets its settings (for example, security group and IAM role) from the instance that you attached. The Auto Scaling group gets settings (for example, Availability Zone and subnet) from the instance that you attached, and has a desired capacity and maximum size of 1.

5. (Optional) To edit the settings for the Auto Scaling group, in the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**. Select the new Auto Scaling group, click **Edit**, change the settings as needed, and then click **Save**.

### To attach an instance to an existing Auto Scaling group using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. (Optional) In the navigation pane, under **Auto Scaling**, click **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, click **Edit**, increase the maximum size, and then click **Save**.
3. In the navigation pane, select **Instances**.
4. Select the instance, click **Actions**, select **Instance Settings**, and then click **Attach to Auto Scaling Group**.
5. In the **Attach to Auto Scaling Group** dialog box, select an **existing Auto Scaling group**, select the instance, and then click **Attach**.
6. If the instance doesn't meet the criteria (for example, if it's not in the same Availability Zone as the Auto Scaling group), you'll get an error message with the details. Click **Close** and try again with an instance that meets the criteria.

## Attaching an Instance Using the AWS CLI

### To attach an instance to an Auto Scaling group using the AWS CLI

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 the group has 2 running instances:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 300,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-asg",
```

```
    "DefaultCooldown": 300,
    "MinSize": 1,
    "Instances": [
      {
        "InstanceId": "i-a5e87793",
        "AvailabilityZone": "us-east-1a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      },
      {
        "InstanceId": "i-a4e87792",
        "AvailabilityZone": "us-east-1a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      }
    ],
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-e4f33493",
    "TerminationPolicies": [
      "Default"
    ],
    "LaunchConfigurationName": "my-lc",
    "CreatedTime": "2014-12-12T23:30:42.611Z",
    "AvailabilityZones": [
      "us-east-1a"
    ],
    "HealthCheckType": "EC2"
  }
]
```

2. Attach an instance to the Auto Scaling group using the following [attach-instances](#) command:

```
aws autoscaling attach-instances --instance-ids i-a8e09d9c --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 to 3, and that there is a new instance, `i-a8e09d9c`:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 300,
      "SuspendedProcesses": [],
      "DesiredCapacity": 3,

```

```
    "Tags": [],
    "EnabledMetrics": [],
    "LoadBalancerNames": [],
    "AutoScalingGroupName": "my-asg",
    "DefaultCooldown": 300,
    "MinSize": 1,
    "Instances": [
      {
        "InstanceId": "i-a8e09d9c",
        "AvailabilityZone": "us-east-1a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      },
      {
        "InstanceId": "i-a5e87793",
        "AvailabilityZone": "us-east-1a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      },
      {
        "InstanceId": "i-a4e87792",
        "AvailabilityZone": "us-east-1a",
        "HealthStatus": "Healthy",
        "LifecycleState": "InService",
        "LaunchConfigurationName": "my-lc"
      }
    ],
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-e4f33493",
    "TerminationPolicies": [
      "Default"
    ],
    "LaunchConfigurationName": "my-lc",
    "CreatedTime": "2014-12-12T23:30:42.611Z",
    "AvailabilityZones": [
      "us-east-1a"
    ],
    "HealthCheckType": "EC2"
  }
]
```

## Detach EC2 Instances From Your Auto Scaling Group

You can remove an instance from an Auto Scaling group. After the instances are detached, you can manage them 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 one. For more information, see [Attach EC2 Instances to Your Auto Scaling Group \(p. 95\)](#).

- Test an Auto Scaling group by creating it using existing instances running your application, and then detach these instances from the Auto Scaling group when your tests are complete.

When you detach instances, you have the option of decrementing the desired capacity for the Auto Scaling group by the number of instances being detached. If you choose not to decrement the capacity, Auto Scaling launches new instances to replace the ones that you detached.

If you detach an instance from an Auto Scaling group that is also registered with a load balancer, the instance is deregistered from the load balancer. If connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete.

### Contents

- [Detaching Instances Using the AWS Management Console \(p. 99\)](#)
- [Detaching Instances Using the AWS CLI \(p. 99\)](#)

Note that 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
- Availability Zone = `us-east-1a`

## Detaching Instances Using the AWS Management Console

Use the following procedure to detach an instance from your Auto Scaling group.

### To detach an instance from an existing Auto Scaling group using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. In the bottom pane, select the **Instances** tab.
5. Select the instance, click **Actions**, and then click **Detach**.
6. In the **Detach Instance** dialog box, select the checkbox if you want Auto Scaling to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Click **Detach Instance**.

## Detaching Instances Using the AWS CLI

Use the following procedure to detach an instance from your Auto Scaling group.

### To detach an instance from an existing Auto Scaling group using the AWS CLI

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 4 running instances:



```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-2a2d8978",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    },
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-5f2e8a0d",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    },
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-a52387f7",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    },
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-f42d89a6",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    }
  ]
}
```

2. Detach an instance and decrement the desired capacity using the following [detach-instances](#) command:

```
aws autoscaling detach-instances --instance-ids i-2a2d8978 --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

3. Verify that the instance is detached using the following `as-describe-auto-scaling-instances` command:

```
aws autoscaling describe-auto-scaling-instances
```

The following example response shows that there are now 3 running instances:

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
```

```
    "InstanceId": "i-5f2e8a0d",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  }
  {
    "AvailabilityZone": "us-east-1a",
    "InstanceId": "i-a52387f7",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  }
  {
    "AvailabilityZone": "us-east-1a",
    "InstanceId": "i-f42d89a6",
    "AutoScalingGroupName": "my-asg",
    "HealthStatus": "HEALTHY",
    "LifecycleState": "InService",
    "LaunchConfigurationName": "my-lc"
  }
]
}
```

## Merge Your Auto Scaling Groups into a Single Multi-Zone Group

To merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones, rezone one of the single-zone groups into a multi-zone group, and then delete the other groups. This process works for groups with or without a load balancer, as long as the new multi-zone group is in one of the same Availability Zones as the original single-zone groups.

The following examples assume that you have two identical groups in two different Availability Zones, `us-west-2a` and `us-west-2c`. These two groups share the following specifications:

- Minimum size = 2
- Maximum size = 5
- Desired capacity = 3

### Merge Zones Using the AWS CLI

Use the following procedure to merge `my-group-a` and `my-group-c` into a single group that covers both `us-west-2a` and `us-west-2c`.

#### To merge separate single-zone groups into a single multi-zone group

1. Use the following [update-auto-scaling-group](#) command to add the `us-west-2c` Availability Zone to the supported Availability Zones for `my-group-a` and increase the maximum size of this group to allow for the instances from both single-zone groups:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-group-a --availability-zones "us-west-2a" "us-west-2c" --max-size 10 --min-size 4
```

2. Use the following [set-desired-capacity](#) command to increase the size of my-group-a:

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-group-a --desired-capacity 6
```

3. (Optional) Use the following [describe-auto-scaling-groups](#) command to verify that my-group-a is at its new size:

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

4. Use the following [update-auto-scaling-group](#) command to remove the instances from my-group-c:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-group-c --min-size 0 --max-size 0
```

5. (Optional) Use the following [describe-auto-scaling-groups](#) command to verify that no instances remain in my-group-c:

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

The following is example output:

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 300,
      "SuspendedProcesses": [],
      "DesiredCapacity": 0,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-group-c",
      "DefaultCooldown": 300,
      "MinSize": 0,
      "Instances": [],
      "MaxSize": 0,
      "VPCZoneIdentifier": "null",
      "TerminationPolicies": [
        "Default"
      ],
      "LaunchConfigurationName": "my-lc",
      "CreatedTime": "2015-02-26T18:24:14.449Z",
      "AvailabilityZones": [
        "us-west-2c"
      ],
    }
  ]
}
```

```
        "HealthCheckType": "EC2"  
    }  
]  
}
```

6. Use the `delete-auto-scaling-group` command to delete `my-group-c`:

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-group-  
c
```

## Temporarily Removing Instances

You can put instances that are currently in service into a standby state. Instances in this state do not actively handle application traffic, but remain a part of the Auto Scaling group.

By default, Auto Scaling decrements the desired capacity of your Auto Scaling group for every instance put into a standby state. When you return the instance to service, Auto Scaling increments the desired capacity accordingly. This prevents Auto Scaling from launching additional instances while you have instances on standby. You can change this behavior so that Auto Scaling launches additional instances to replace the instances.

### Important

You are billed for any instances in your Auto Scaling group—regardless of whether the instance is in service or on standby.

When you return the instances back to service, Auto Scaling detects that you have more instances than you need, and applies any termination policies to reduce the size of your Auto Scaling group.

### Contents

- [Troubleshooting Instances in an Auto Scaling Group \(p. 103\)](#)
- [Updating or Modifying Instances in an Auto Scaling Group \(p. 107\)](#)

## Troubleshooting Instances in an Auto Scaling Group

If you want to troubleshoot an instance that is currently in service, put it into a `Standby` state. By putting the instance into a `Standby` state, you can make changes to the instance and then return it to service.

When put an instance into a `Standby` state, you must decide whether you want Auto Scaling to launch a replacement instance. If you don't want a replacement instance, Auto Scaling decrements the desired capacity for the Auto Scaling group when you put an instance in a `Standby` state and increments the desired capacity when you put the instance back in the `InService` state. Otherwise, Auto Scaling launches an additional instance to replace the instance moved into the `Standby` state and follows the group's termination policy when you put the instance back in the `InService` state.

The process described in the following procedures requires that an instance is currently in service (it has a status of `InService`.) If an instance is already starting to terminate—for example, because it failed an Amazon EC2 health check—you won't be able to return it to service. You can, however, put the instance in a `Terminating:Wait` state to analyze why the instance failed. For more information, see [Analyzing an Instance Before Termination \(p. 70\)](#).

## Contents

- [Troubleshooting Instances Using the AWS Management Console](#) (p. 104)
- [Troubleshooting Instances Using the AWS CLI](#) (p. 104)

## Troubleshooting Instances Using the AWS Management Console

The following steps demonstrate the general process for troubleshooting an instance that is currently in service.

### To troubleshoot an instance that is currently in service using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. In the bottom pane, select the **Instances** tab.
5. Select the instance, click **Actions**, and then click **Set to Standby**.
6. In the **Set to Standby** dialog box, select the checkbox if you want Auto Scaling to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Click **Set to Standby**.
7. Connect to the instance and review logs or run diagnostics as needed.

For more information, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

8. Select the instance, click **Actions**, and then click **Set to InService**. In the **Set to InService** dialog box, click **Set to InService**.

## Troubleshooting Instances Using the AWS CLI

The following steps demonstrate the general process for troubleshooting an instance that is currently in service.

### To troubleshoot an instance that is currently in service using the AWS CLI

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.

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-e116ddb3",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    },
    ...
  ]
}
```

```
    ]  
  }
```

2. 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 Auto Scaling does not launch a replacement instance.

```
aws autoscaling enter-standby --instance-ids i-e116ddb3 --auto-scaling-group-  
name my-asg --should-decrement-desired-capacity
```

The following is an example response:

```
{  
  "Activities": [  
    {  
      "Description": "Moving EC2 instance to Standby: i-e116ddb3",  
      "AutoScalingGroupName": "my-asg",  
      "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",  
      "Details": "{\"Availability Zone\":\"us-east-1a\"}",  
      "StartTime": "2014-12-15T21:31:26.150Z",  
      "Progress": 50,  
      "Cause": "At 2014-12-15T21:31:26Z instance i-e116ddb3 was moved  
to standby          in response to a user request, shrinking the capacity from 2  
to 1.",  
      "StatusCode": "InProgress"  
    }  
  ]  
}
```

3. (Optional) Verify that the instance is in a Standby state using the following [describe-auto-scaling-instances](#) command:

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-e116ddb3
```

The following is an example response. Notice that the status of the instance is now Standby.

```
{  
  "AutoScalingInstances": [  
    {  
      "AvailabilityZone": "us-east-1a",  
      "InstanceId": "i-e116ddb3",  
      "AutoScalingGroupName": "my-asg",  
      "HealthStatus": "HEALTHY",  
      "LifecycleState": "Standby",  
      "LaunchConfigurationName": "my-lc"  
    }  
  ]  
}
```

4. Connect to the instance and review logs or run diagnostics as needed.

For more information, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

5. Put the instance back in service using the following `exit-standby` command:

```
aws autoscaling exit-standby i-e116ddb3 --auto-scaling-group my-asg
```

The following is an example response:

```
{
  "Activities": [
    {
      "Description": "Moving EC2 instance out of Standby: i-e116ddb3",
      "AutoScalingGroupName": "my-asg",
      "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
      "Details": "{\"Availability Zone\":\"us-east-1a\"}",
      "StartTime": "2014-12-15T21:46:14.678Z",
      "Progress": 30,
      "Cause": "At 2014-12-15T21:46:14Z instance i-e116ddb3 was moved out of standby in response to a user request, increasing the capacity from 1 to 2.",
      "StatusCode": "PreInService"
    }
  ]
}
```

6. (Optional) Verify that the instance is back in service using the following `as-describe-auto-scaling-instances` command:

```
as-describe-auto-scaling-instances --instance-ids i-e116ddb3
```

The following is an example response. Notice that the status of the instance is back to `InService`.

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-e116ddb3",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    }
  ]
}
```

## Updating or Modifying Instances in an Auto Scaling Group

You can assign a new launch configuration to an Auto Scaling group at any time. This practice is common when you want new instances to use an updated configuration. However, changing the launch configuration for an Auto Scaling group does not update any instances currently in service. You can update these instances by putting them into a `Standby` state, updating the software, and then putting the instances back in service.

When put an instance into a `Standby` state, you must decide whether you want Auto Scaling to launch a replacement instance. If you don't want a replacement instance, Auto Scaling decrements the desired capacity for the Auto Scaling group when you put an instance in a `Standby` state and increments the desired capacity when you put the instance back in the `InService` state. Otherwise, Auto Scaling launches an additional instance to replace the instance moved into the `Standby` state and follows the group's termination policy when you put the instance back in the `InService` state.

### Contents

- [Updating an Instance Using the AWS Management Console \(p. 107\)](#)
- [Updating an Instance Using the AWS CLI \(p. 107\)](#)

## Updating an Instance Using the AWS Management Console

The following procedure demonstrates the general process for updating an instance that is currently in service.

### To update software on an instance using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group from the list.
4. In the bottom pane, select the **Instances** tab.
5. Select the instance, click **Actions**, and then click **Set to Standby**.
6. In the **Set to Standby** dialog box, select the checkbox if you want Auto Scaling to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Click **Set to Standby**.
7. Connect to the instance and update the software as needed.

For more information, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

8. Select the instance, click **Actions**, and then click **Set to InService**. In the **Set to InService** dialog box, click **Set to InService**.

## Updating an Instance Using the AWS CLI

The following procedure demonstrates the general process for updating an instance that is currently in service.

### To update software on an instance using the AWS CLI

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:

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-5b73d709",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "InService",
      "LaunchConfigurationName": "my-lc"
    },
    ...
  ]
}
```

2. 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 Auto Scaling does not launch a replacement instance.

```
aws autoscaling enter-standby --instance-ids i-5b73d709 --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

The following is an example response:

```
{
  "Activities": [
    {
      "Description": "Moving EC2 instance to Standby: i-5b73d709",
      "AutoScalingGroupName": "my-asg",
      "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
      "Details": "{\"Availability Zone\":\"us-east-1a\"}",
      "StartTime": "2014-12-15T21:31:26.150Z",
      "Progress": 50,
      "Cause": "At 2014-12-15T21:31:26Z instance i-5b73d709 was moved to standby in response to a user request, shrinking the capacity from 4 to 3.",
      "StatusCode": "InProgress"
    }
  ]
}
```

3. (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-5b73d709
```

The following is an example response. Notice that the status of the instance is now Standby.

```
{
  "AutoScalingInstances": [
    {
      "AvailabilityZone": "us-east-1a",
      "InstanceId": "i-5b73d709",
      "AutoScalingGroupName": "my-asg",
      "HealthStatus": "HEALTHY",
      "LifecycleState": "Standby",
      "LaunchConfigurationName": "my-lc"
    }
  ]
}
```

4. Connect to the instance and update the software as needed.

For more information, see [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* or [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

5. Put the instance back in service using the following `exit-standby` command:

```
aws autoscaling exit-standby --instance-ids i-5b73d709 --auto-scaling-group-name my-asg
```

The following is an example response:

```
{
  "Activities": [
    {
      "Description": "Moving EC2 instance out of Standby: i-5b73d709",
      "AutoScalingGroupName": "my-asg",
      "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
      "Details": "{\"Availability Zone\":\"us-east-1a\"}",
      "StartTime": "2014-12-15T21:46:14.678Z",
      "Progress": 30,
      "Cause": "At 2014-12-15T21:46:14Z instance i-5b73d709 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-5b73d709
```

The following is an example response. Notice that the status of the instance is back to `InService`.

```
{
  "AutoScalingInstances": [
```

```
{
  "AvailabilityZone": "us-east-1a",
  "InstanceId": "i-5b73d709",
  "AutoScalingGroupName": "my-asg",
  "HealthStatus": "HEALTHY",
  "LifecycleState": "InService",
  "LaunchConfigurationName": "my-lc"
}
```

## Suspend and Resume Auto Scaling Processes

Auto Scaling enables you to suspend and then resume one or more of the Auto Scaling processes in your Auto Scaling group. This can be very useful when you want to investigate a configuration problem or other issue with your web application and then make changes to your application, without triggering the Auto Scaling process.

Auto Scaling might suspend processes for Auto Scaling groups that repeatedly fail to launch instances. This is known as an *administrative suspension*, and 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 suspended for administrative reasons.

### Contents

- [Auto Scaling Processes \(p. 110\)](#)
- [Suspend and Resume Processes Using the AWS CLI \(p. 111\)](#)

## Auto Scaling Processes

Auto Scaling supports the following processes:

### Launch

Adds a new EC2 instance to the group.

#### Warning

If you suspend Launch, this disrupts other processes.

### Terminate

Removes an EC2 instance from the group.

#### Warning

If you suspend Terminate, this disrupts other processes.

### HealthCheck

Checks the health of the instances. Auto Scaling marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing tells Auto Scaling that the instance is unhealthy. This process can override the health status of an instance that you set manually.

### ReplaceUnhealthy

Terminates instances that are marked as unhealthy and subsequently creates new instances to replace them. This process works with the HealthCheck process, and uses both the Terminate and Launch processes.

### AZRebalance

Balances the number of EC2 instances in the group across the Availability Zones in the region. If you remove an Availability Zone from your Auto Scaling group or an Availability Zone otherwise

becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone before terminating the unhealthy or unavailable instances. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the instances evenly across the Availability Zones for the group.

Note that if you suspend `AZRebalance` and a scale out or scale in event occurs, Auto Scaling still tries to balance the Availability Zones. For example, during scale out, Auto Scaling launches the instance in the Availability Zone with the fewest instances.

If you suspend `Launch`, `AZRebalance` neither launches new instances nor terminates existing instances. This is because `AZRebalance` terminates instances only after launching the replacement instances. If you suspend `Terminate`, your Auto Scaling group can grow up to ten percent larger than its maximum size, because Auto Scaling allows this temporarily during rebalancing activities. If Auto Scaling cannot terminate instances, your Auto Scaling group could remain above its maximum size until you resume the `Terminate` process.

#### `AlarmNotification`

Accepts notifications from CloudWatch alarms that are associated with the group.

If you suspend `AlarmNotification`, Auto Scaling does not automatically execute policies that would be triggered by an alarm. If you suspend `Launch` or `Terminate`, Auto Scaling would not be able to execute scale-out or scale-in policies, respectively.

#### `ScheduledActions`

Performs scheduled actions that you create.

If you suspend `Launch` or `Terminate`, scheduled actions that involve launching or terminating instances are affected.

#### `AddToLoadBalancer`

Adds instances to the load balancer when they are launched.

If you suspend `AddToLoadBalancer`, Auto Scaling launches the instances but does not add them to the load balancer. If you resume the `AddToLoadBalancer` process, Auto Scaling resumes adding instances to the load balancer when they are launched. However, Auto Scaling does not add the instances that were launched while this process was suspended. You must register those instances manually. For more information, see [Registering Your Amazon EC2 Instances with Your Load Balancer](#) in the *Elastic Load Balancing Developer Guide*.

## Suspend and Resume Processes Using the AWS CLI

You can suspend and resume individual processes (using the `--scaling-processes` option) or all processes (omit the `--scaling-processes` option).

### To suspend all processes for an Auto Scaling group

Use the `suspend-processes` command as follows:

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

### To resume all suspended processes for an Auto Scaling group

After concluding your investigation, use the `resume-processes` command as follows:

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

# Shut Down Auto Scaling Processes Using the AWS CLI

To completely shut down the Auto Scaling process, complete the following tasks using the AWS CLI.

## Tasks

- [Delete Your Auto Scaling Group](#) (p. 112)
- (Optional) [Delete the Launch Configuration](#) (p. 112)
- (Optional) [Delete the Load Balancer](#) (p. 112)
- (Optional) [Delete CloudWatch Alarms](#) (p. 113)

## Delete Your Auto Scaling Group

You can delete your Auto Scaling group if it has no running instances. To ensure that your Auto Scaling group has no running instances, set its minimum size and maximum size to zero using the following [update-auto-scaling-group](#) command:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg -  
--max-size 0 --min-size 0
```

You can verify that your Auto Scaling group has no running instances using the following [describe-auto-scaling-groups](#) command:

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

Auto Scaling might report that the instances are in the `Terminating` state because the termination process can take a few minutes.

After the instances have terminated, use the following [delete-auto-scaling-group](#) command to delete the Auto Scaling group:

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

## (Optional) Delete the Launch Configuration

Note that you can skip this step if you want to keep the launch configuration for future use.

To delete the launch configuration associated with the Auto Scaling group, use the following [delete-launch-configuration](#) command:

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-lc
```

## (Optional) Delete the Load Balancer

Note that you can skip this step if your Auto Scaling group is not registered with an Elastic Load Balancing load balancer or you want to keep the load balancer for future use.

To delete a load balancer, use the following [delete-load-balancer](#) command:

```
aws elb delete-load-balancer my-load-balancer
```

## (Optional) Delete CloudWatch Alarms

Note that you can skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you want to keep the CloudWatch alarms for future use.

To delete CloudWatch alarms, use the [delete-alarms](#) command. For example, use the following command to delete the `AddCapacity` and `RemoveCapacity` alarms:

```
aws cloudwatch delete-alarms --alarm-name AddCapacity RemoveCapacity
```

# Monitoring Your Auto Scaling Instances

---

Auto Scaling instances send metrics to Amazon CloudWatch. Instance metrics are the metrics that an individual EC2 instance sends to CloudWatch. Instance metrics are the same metrics available for any EC2 instance, whether or not it is in an Auto Scaling group.

CloudWatch offers basic or detailed monitoring. Basic monitoring sends aggregated data about each instance to CloudWatch every five minutes. Detailed monitoring offers more frequent aggregated data by sending data from each instance every minute.

## Contents

- [Amazon CloudWatch Alarms \(p. 114\)](#)
- [Activating Detailed Instance Monitoring for Auto Scaling \(p. 115\)](#)
- [Activating Basic Instance Monitoring for Auto Scaling \(p. 115\)](#)
- [Auto Scaling Group Metrics \(p. 116\)](#)
- [Health Checks \(p. 117\)](#)
- [Getting Notifications When Your Auto Scaling Group Changes \(p. 119\)](#)
- [Logging Auto Scaling API Calls By Using AWS CloudTrail \(p. 123\)](#)

## Amazon CloudWatch Alarms

A CloudWatch *alarm* is an object that monitors a single metric over a specific period. A metric is a variable that you want to monitor, such as average CPU usage of the EC2 instances, or incoming network traffic from many different EC2 instances. The alarm changes its state when the value of the metric breaches a defined range and maintains the change for a specified number of periods.

An alarm has three possible states:

- **OK**— When the value of the metric remains within the range that you've specified.
- **ALARM**— When the value of the metric goes out of the range that you've specified and remains outside of the range for a specified time duration.
- **INSUFFICIENT\_DATA**— When the metric is not yet available or not enough data is available for the metric to determine the alarm state.

When the alarm changes to the ALARM state and remains in that state for a number of periods, it invokes one or more actions. The actions can be a message sent to an Auto Scaling group to change the desired capacity of the group.

You configure an alarm by identifying the metrics to monitor. For example, you can configure an alarm to watch over the average CPU usage of the EC2 instances in an Auto Scaling group.

You must use CloudWatch to identify metrics and create alarms. For more information, see [Creating CloudWatch Alarms](#) in the *Amazon CloudWatch Developer Guide*.

## Activating Detailed Instance Monitoring for Auto Scaling

To enable detailed instance monitoring for a new Auto Scaling group, you don't need to take any extra steps. One of your first steps when creating an Auto Scaling group is to create a launch configuration. Each launch configuration contains a flag named `InstanceMonitoring.Enabled`. The default value of this flag is `true`, so you don't need to set this flag manually if you want detailed monitoring.

If you have an Auto Scaling group for which you have explicitly selected basic monitoring, the switch to detailed monitoring involves several steps, especially if you have CloudWatch alarms configured to scale the group automatically.

### To switch to detailed instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag enabled. If you are using the [AWS CLI](#), create a launch configuration with the `--instance-monitoring` option.
2. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration that you created in the previous step. Auto Scaling enables detailed monitoring for new instances that it creates.
3. Choose one of the following actions to deal with all existing EC2 instances in the Auto Scaling group:

Task	Action
Preserve existing instances	Call <code>MonitorInstances</code> from the Amazon EC2 API for each existing instance to enable detailed monitoring.
Terminate existing instances	Call <code>TerminateInstanceInAutoScalingGroup</code> from the Auto Scaling API for each existing instance. Auto Scaling uses the updated launch configuration to create replacement instances with detailed monitoring enabled.

4. If you have CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the CloudWatch API to update each alarm so that the alarm's period value is set to 60 seconds.

## Activating Basic Instance Monitoring for Auto Scaling

To create a new Auto Scaling group with basic monitoring instead of detailed monitoring, associate your new Auto Scaling group with a launch configuration that has the `InstanceMonitoring.Enabled` flag set to `false`.



### To switch to basic instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag disabled. If you are using the CLI, create a launch configuration with the `--monitoring-disabled` option.
2. If you previously enabled group metrics with a call to `EnableMetricsCollection`, call `DisableMetricsCollection` on your Auto Scaling group to disable collection of all group metrics. For more information, see [Auto Scaling Group Metrics \(p. 116\)](#).
3. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration that you created in the previous step. Auto Scaling disables detailed monitoring for new instances that it creates.
4. Choose one of the following actions to deal with all existing EC2 instances in the Auto Scaling group:

Task	Action
Preserve existing instances	Call <code>UnmonitorInstances</code> from the Amazon EC2 API for each existing instance to disable detailed monitoring.
Terminate existing instances	Call <code>TerminateInstanceInAutoScalingGroup</code> from the Auto Scaling API for each existing instance. Auto Scaling uses the updated launch configuration to create replacement instances with detailed monitoring disabled.

5. If you have CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the CloudWatch API to update each alarm so that the alarm's period value is set to 300 seconds.

#### Important

If you do not update your alarms to match the five-minute data aggregations, your alarms continue to check for statistics every minute and might find no data available for as many as four out of every five periods.

For more information about instance metrics for EC2 instances, see the [Amazon CloudWatch Developer Guide](#).

## Auto Scaling Group Metrics

Group metrics are metrics that Auto Scaling group sends to CloudWatch to describe the group rather than any of its instances. If you enable group metrics, Auto Scaling sends aggregated data to CloudWatch every minute. If you disable group metrics, Auto Scaling does not send any group metrics data to CloudWatch.

### To enable group metrics

1. Enable detailed instance monitoring for the Auto Scaling group by setting the `InstanceMonitoring.Enabled` flag in the Auto Scaling group's launch configuration. For more information, see [Monitoring Your Auto Scaling Instances \(p. 114\)](#).
2. Call `EnableMetricsCollection`. Alternatively, you can use the equivalent `aws autoscaling enable-metrics-collection` command that is part of the AWS CLI.

## Auto Scaling Group Metrics Table

You may enable or disable each of the following metrics, separately.

Metric	Description
GroupMinSize	The minimum size of the Auto Scaling group.
GroupMaxSize	The maximum size of the Auto Scaling group.
GroupDesiredCapacity	The number of instances that the Auto Scaling group attempts to maintain.
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.
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.
GroupStandbyInstances	The number of instances that are in a <code>Standby</code> state. Instances in this state are still running but are not actively in service. This metric is not included by default; you must request it specifically.
GroupTerminatingInstances	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.
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.

## Dimensions for Auto Scaling Group Metrics

The only dimension that Auto Scaling sends to CloudWatch is the name of the Auto Scaling group. This means that all available statistics are filtered by Auto Scaling group name.

## Health Checks

Auto Scaling periodically performs health checks on the instances in your group and replaces instances that fail these checks. By default, these health checks use the results of EC2 instance status checks to determine the health of an instance. If you use a load balancer with your Auto Scaling group, you can optionally choose to include the results of Elastic Load Balancing health checks.

Auto Scaling marks an instance as unhealthy if the instance status is any state other than `running`, the system status is `impaired`, or Elastic Load Balancing reports the instance state as `OutOfService`. For more information about Amazon EC2 status checks, see [Monitoring the Status of your Instances](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about Elastic Load Balancing health checks, see [Elastic Load Balancing Health Check](#) in the *Elastic Load Balancing Developer Guide*.

You can customize the health check conducted by your Auto Scaling group by specifying additional checks or by having your own health check system and then sending the instance's health information directly from your system to Auto Scaling. For more information, see [Set the Health State of An Instance \(p. 118\)](#).

After an instance is marked unhealthy because of an Amazon EC2 or Elastic Load Balancing health check, it is scheduled for replacement. For more information, see [Maintaining a Fixed Number of EC2 Instances in Your Auto Scaling Group \(p. 29\)](#) and [Add an Elastic Load Balancing Health Check to your Auto Scaling Group \(p. 91\)](#).

## Set the Health State of An Instance

If you have your own health check system, you can use the information from your health check system to determine how to set the health state of the instances in the Auto Scaling group.

### Setting the Instance Health State Using the AWS CLI

You can specify the health state of an instance in your Auto Scaling group. Note that if you set the health state of an instance to `Unhealthy`, Auto Scaling terminates the instance.

#### To configure the health state of an instance using the AWS CLI

1. Use the following `set-instance-health` command to set the health status of the specified instance to `Unhealthy`:

```
aws autoscaling set-instance-health --instance-id i-123abc45d --health-status  
Unhealthy
```

2. 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 instance status is `Unhealthy` and that it is terminating:

```
{  
  "AutoScalingGroups": [  
    {  
      ....  
      "Instances": [  
        {  
          "InstanceId": "i-123abc45d",  
          "AvailabilityZone": "us-east-1a",  
          "HealthStatus": "Unhealthy",  
          "LifecycleState": "Terminating",  
          "LaunchConfigurationName": "my-lc"  
        },  
        ...  
      ]  
    }  
  ]  
}
```

# Getting Notifications When Your Auto Scaling Group Changes

When you use Auto Scaling to scale your applications automatically, you want to know when Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. You can configure your Auto Scaling group to send a notification, whenever the Auto Scaling group changes.

If configured, the Auto Scaling group uses Amazon Simple Notification Service (Amazon SNS) to send the notifications. Amazon SNS coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. Amazon SNS can deliver notifications as HTTP or HTTPS POST, email (SMTP, either plain-text or in JSON format), or as a message posted to an Amazon SQS queue. For more information, see [What Is Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

To configure your Auto Scaling group to send email notifications whenever your Auto Scaling group changes, complete the following tasks.

## Tasks

- [Configure Amazon SNS](#) (p. 119)
- [Configure Your Auto Scaling Group to Send Notifications](#) (p. 120)
- [Test the Notification Configuration](#) (p. 121)
- [Verify That You Received Notification of the Scaling Event](#) (p. 121)
- [Delete the Notification Configuration](#) (p. 123)

## Configure Amazon SNS

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.

For more information, see [Create a Topic](#) in the *Amazon Simple Notification Service Developer Guide*.

### Subscribe to the Amazon SNS Topic

To receive notifications your Auto Scaling group sends to the topic, you must subscribe an endpoint to the topic. In this procedure, for the **Endpoint** field, specify the email address where you want to receive the notifications from Auto Scaling.

For more information, see [Subscribe to a 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 you open the email from AWS Notifications and click the link to confirm the subscription before you continue with the next step.

You will receive an acknowledgement 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.

When you configure your Auto Scaling group to send email notifications, you must specify the notification types for the Auto Scaling group. Auto Scaling supports sending Amazon SNS notifications when the following events occur:

Notification type	Event
<code>autoscaling:EC2_INSTANCE_LAUNCH</code>	Successful instance launch
<code>autoscaling:EC2_INSTANCE_LAUNCH_ERROR</code>	Failed instance launch
<code>autoscaling:EC2_INSTANCE_TERMINATE</code>	Successful instance termination
<code>autoscaling:EC2_INSTANCE_TERMINATE_ERROR</code>	Failed instance termination
<code>autoscaling:TEST_NOTIFICATION</code>	Validated a configured SNS topic (as a result of calling the <a href="#">PutNotificationConfiguration</a> action)

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.

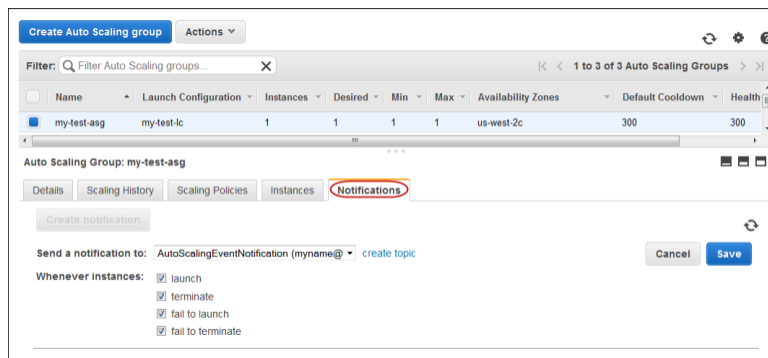
### Contents

- [Configure Notifications Using the Console](#) (p. 120)
- [Configure Notifications Using the AWS CLI](#) (p. 121)

## Configure Notifications Using the Console

To configure Amazon SNS notifications for your Auto Scaling group using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group from the list.
4. The bottom pane displays the details of your Auto Scaling group. In the bottom pane, click **Notifications** tab.
5. Click **Create notification**.
6. In the **Create notifications** pane, do the following:
  - a. Click the **Send a notification to:** field and select your SNS topic.
  - b. In the **Whenever instances** event list, select the events to send the notifications for.
  - c. Click **Save**.



## Configure Notifications Using the AWS CLI

To configure Amazon SNS notifications for your Auto Scaling group

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" "auto
scaling:EC2_INSTANCE_TERMINATE"
```

## Test the Notification Configuration

To cause the changes that generate notifications, update the Auto Scaling group by changing the desired capacity of the Auto Scaling group; for example, from 1 instance to 2 instances. After Auto Scaling launches the EC2 instance, you'll receive the email notification with a few minutes.

To change the desired capacity using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. In the bottom pane, click the **Details** tab.
5. Click **Edit**.
6. In the **Desired** field, enter 2, and then click **Save**.

To change the desired capacity using the AWS CLI

Use the following [set-desired-capacity](#) command:

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg --desired-
capacity 2
```

## Verify That You Received Notification of the Scaling Event

Check your email for a message from Amazon SNS and open the email. After you receive notification of a scaling event for your Auto Scaling group, you can confirm the scaling event by looking at the description

## Auto Scaling Developer Guide

### Verify That You Received Notification of the Scaling Event

---

of your Auto Scaling group. You'll need information from the notification email, such as the ID of the instance that was launched or terminated.

#### To verify that your Auto Scaling group has launched new instance using the console

1. Select your Auto Scaling group.
2. In the bottom pane, click the **Scaling History** tab.
3. Check the **Status** column to determine the current status of your instance. For example, if the notification indicates that an instance has launched, click the refresh button to verify that the status of the launch activity is **Successful**.
4. On the **Instances** tab, you can view the current **Lifecycle** state of the instance whose ID you received in the notification email. After a new instance starts, its lifecycle state changes to **InService**.

#### To verify that your Auto Scaling group has launched new instance using the AWS CLI

Use the following [describe-auto-scaling-groups](#) command to confirm that the size of your Auto Scaling group has changed:

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

The following example output shows that the group has two instances. Check for the instance whose ID you received in the notification email.

```
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupARN": "arn",
      "HealthCheckGracePeriod": 0,
      "SuspendedProcesses": [],
      "DesiredCapacity": 2,
      "Tags": [],
      "EnabledMetrics": [],
      "LoadBalancerNames": [],
      "AutoScalingGroupName": "my-asg",
      "DefaultCooldown": 300,
      "MinSize": 1,
      "Instances": [
        {
          "InstanceId": "i-d95eb0d4",
          "AvailabilityZone": "us-west-2b",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "my-lc"
        },
        {
          "InstanceId": "i-13d7dc1f",
          "AvailabilityZone": "us-west-2a",
          "HealthStatus": "Healthy",
          "LifecycleState": "InService",
          "LaunchConfigurationName": "my-lc"
        }
      ],
      "MaxSize": 5,
      "VPCZoneIdentifier": null,
      "TerminationPolicies": [
```

```
        "Default"
      ],
      "LaunchConfigurationName": "my-lc",
      "CreatedTime": "2015-03-01T16:12:35.608Z",
      "AvailabilityZones": [
        "us-west-2b",
        "us-west-2a"
      ],
      "HealthCheckType": "EC2"
    }
  ]
}
```

## Delete the Notification Configuration

You can delete your Auto Scaling notification configuration at any time.

### To delete Auto Scaling notification configuration using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. In the bottom pane, click the **Notifications** tab.
5. In the notifications pane, click the **Delete** button next to the notification.

### To delete Auto Scaling notification configuration using the AWS CLI

Use the following `delete-notification-configuration` command:

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

For information about deleting the Amazon SNS topic associated with your Auto Scaling group, and also deleting all the subscriptions to that topic, see [Clean Up](#) in the *Amazon Simple Notification Service Developer Guide*.

## Logging Auto Scaling API Calls By Using AWS CloudTrail

Auto Scaling is integrated with CloudTrail, a service that captures API calls made by or on behalf of Auto Scaling in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the Auto Scaling console or from the Auto Scaling API. Using the information collected by CloudTrail, you can determine what request was made to Auto Scaling, the source IP address from which the request was made, who made the request, when it was made, and so on. For more information about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).



## Auto Scaling Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to Auto Scaling actions are tracked in log files. Auto Scaling records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the Auto Scaling actions are logged and are documented in the [Auto Scaling API Reference](#). For example, calls to the **CreateLaunchConfiguration**, **DescribeAutoScalingGroup**, and **UpdateAutoScalingGroup** actions generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with account or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the **userIdentity** field in the [CloudTrail Event Reference](#) section in the *AWS CloudTrail User Guide*.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see [Configuring Amazon SNS Notifications](#) in the *AWS CloudTrail User Guide*.

You can also aggregate Auto Scaling log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see [Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket](#) in the *AWS CloudTrail User Guide*.

## Understanding Auto Scaling Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that demonstrates the **CreateLaunchConfiguration** action.

```
{
  "Records": [
    {
      "eventVersion": "1.01",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/iamUser1",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "iamUser1"
      },
      "eventTime": "2014-06-24T16:53:14Z",
      "eventSource": "autoscaling.amazonaws.com",
      "eventName": "CreateLaunchConfiguration",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",

```

```
    "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
    "requestParameters": {
      "imageId": "ami-2f726546",
      "instanceType": "m1.small",
      "launchConfigurationName": "launch_configuration_1"
    },
    "responseElements": null,
    "requestID": "07a1becf-fbc0-11e3-bfd8-a5209058e7bb",
    "eventID": "ad30abf7-57db-4a6d-93fa-13deb1fd4cff"
  },
  ...additional entries
]
}
```

The following example shows a CloudTrail log entry that demonstrates the **DescribeAutoScalingGroups** action.

```
{
  "Records": [
    {
      "eventVersion": "1.01",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/iamUser1",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "iamUser1"
      },
      "eventTime": "2014-06-23T23:20:56Z",
      "eventSource": "autoscaling.amazonaws.com",
      "eventName": "DescribeAutoScalingGroups",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
      "requestParameters": {
        "maxRecords": 20
      },
      "responseElements": null,
      "requestID": "0737e2ea-fb2d-11e3-bfd8-a5209058e7bb",
      "eventID": "0353fb04-281e-47d9-93bb-588bf2256538"
    },
    ...additional entries
  ]
}
```

The following example shows a CloudTrail log entry that demonstrates the **UpdateAutoScalingGroups** action.

```
{
  "Records": [
    {
      "eventVersion": "1.01",
      "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/iamUser1",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "iamUser1"
    },
    "eventTime": "2014-06-24T16:54:46Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "UpdateAutoScalingGroup",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
    "requestParameters": {
        "maxSize": 8,
        "minSize": 1,
        "autoScalingGroupName": "asg1"
    },
    "responseElements": null,
    "requestID": "3ed07c03-fbc0-11e3-bfd8-a5209058e7bb",
    "eventID": "b52ca0aa-5199-4873-a546-55f7c896a4ce"
},
...additional entries
]
}
```

# Troubleshooting Auto Scaling

---

Amazon Web Services provides specific and descriptive errors to help you troubleshoot Auto Scaling problems. You can find the error messages in the description of the Auto Scaling activities.

## Contents

- [Retrieving an Error Message \(p. 127\)](#)
- [Troubleshooting Auto Scaling: EC2 Instance Launch Failures \(p. 129\)](#)
- [Troubleshooting Auto Scaling: AMI Issues \(p. 132\)](#)
- [Troubleshooting Auto Scaling: Load Balancer Issues \(p. 134\)](#)
- [Troubleshooting Auto Scaling: Capacity Limits \(p. 136\)](#)

## Retrieving an Error Message

To retrieve an error message from the description of Auto Scaling activities, use the [describe-scaling-activities](#) command as follows:

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

The following is an example response, where `StatusCode` contains the current status of the activity and `StatusMessage` contains the error message:

```
{
  "Activities": [
    {
      "Description": "Launching a new EC2 instance: i-4ba0837f",
      "AutoScalingGroupName": "my-asg",
      "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
      "Details": "{\"Availability Zone\":\"us-west-2c\"}",
      "StartTime": "2013-08-19T20:53:29.930Z",
      "Progress": 100,
      "EndTime": "2013-08-19T20:54:02Z",
      "Cause": "At 2013-08-19T20:53:25Z a user request created an Auto
ScalingGroup...",
      "StatusCode": "Failed",
      "StatusMessage": "The image id 'ami-4edb0327' does not exist."
    }
  ]
}
```

```
Launching EC2 instance failed."  
    }  
  ]  
}
```

The following tables list the types of error messages and provide links to the troubleshooting resources that you can use to troubleshoot your Auto Scaling issues.

### EC2 Instance Launch Failures

Issue	Error Message
Auto Scaling group	<a href="#">AutoScalingGroup &lt;Auto Scaling group name&gt; not found. (p. 131)</a>
Availability Zone	<a href="#">The requested Availability Zone is no longer supported. Please retry your request ..... (p. 131)</a>
AWS account	<a href="#">You are not subscribed to this service. Please see http://aws.amazon.com. (p. 131)</a>
Block device mapping	<a href="#">Invalid device name upload. Launching EC2 instance failed. (p. 131)</a>
Block device mapping	<a href="#">Value (&lt;name associated with the instance storage device&gt;) for parameter virtualName is invalid... (p. 132)</a>
Block device mapping	<a href="#">EBS block device mappings not supported for instance-store AMIs. (p. 132)</a>
Instance type and Availability Zone	<a href="#">Your requested instance type (&lt;instance type&gt;) is not supported in your requested Availability Zone (&lt;instance Availability Zone&gt;).... (p. 131)</a>
Key pair	<a href="#">The key pair &lt;key pair associated with your EC2 instance&gt; does not exist. Launching EC2 instance failed. (p. 130)</a>
Launch configuration	<a href="#">The requested configuration is currently not supported. (p. 130)</a>
Placement group	<a href="#">Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 132)</a>
Security group	<a href="#">The security group &lt;name of the security group&gt; does not exist. Launching EC2 instance failed. (p. 130)</a>

### AMI Issues

Issue	Error Message
AMI ID	<a href="#">The AMI ID &lt;ID of your AMI&gt; does not exist. Launching EC2 instance failed. (p. 133)</a>
AMI ID	<a href="#">AMI &lt;AMI ID&gt; is pending, and cannot be run. Launching EC2 instance failed. (p. 133)</a>
AMI ID	<a href="#">Value (&lt;ami ID&gt;) for parameter virtualName is invalid. (p. 133)</a>
Architecture mismatch	<a href="#">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. 134)</a>

Issue	Error Message
Virtualization type	Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed. (p. 133)

### Load Balancer Issues

Issue	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 134)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 135)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 135)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 135)

### Capacity Limits

Issue	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 136)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>). .... (p. 136)

## Troubleshooting Auto Scaling: EC2 Instance Launch Failures

This page provides information about your EC2 instances that fail to launch with Auto Scaling, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see [Retrieving an Error Message](#) (p. 127).

When your EC2 instances fail to launch, you might get one or more of the following error messages:

### Error Messages

- The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 130)
- The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 130)
- The requested configuration is currently not supported. (p. 130)
- AutoScalingGroup <Auto Scaling group name> not found. (p. 131)
- The requested Availability Zone is no longer supported. Please retry your request ..... (p. 131)
- Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>). .... (p. 131)
- You are not subscribed to this service. Please see <http://aws.amazon.com>. (p. 131)

- [Invalid device name upload. Launching EC2 instance failed.](#) (p. 131)
- [Value \(<name associated with the instance storage device>\) for parameter virtualName is invalid...](#) (p. 132)
- [EBS block device mappings not supported for instance-store AMIs.](#) (p. 132)
- [Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.](#) (p. 132)

## The security group <name of the security group> does not exist. Launching EC2 instance failed.

- **Cause:** The security group specified in your 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 configuration.
  4. Update your Auto Scaling group with the new 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 configuration.
  4. Update your Auto Scaling group with the new launch configuration using the [update-auto-scaling-group](#) command.

## The requested configuration is currently not supported.

- **Cause:** Some fields in your launch configuration might not be currently supported.
- **Solution:**
  1. Create a new launch configuration.
  2. Update your Auto Scaling group with the new launch configuration using the [update-auto-scaling-group](#) command.

## AutoScalingGroup <Auto Scaling group name> not found.

- **Cause:** The Auto Scaling group might have been deleted.
- **Solution:** Create a new Auto Scaling group.

## 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 associated with your launch configuration 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.

## You are not subscribed to this service. Please see <http://aws.amazon.com>.

- **Cause:** Your AWS account might have expired.
- **Solution:** Go to <http://aws.amazon.com> and click **Sign Up Now** to open a new account.

## Invalid device name upload. Launching EC2 instance failed.

- **Cause:** The block device mappings in your launch configuration might contain block device names that are not available or currently not supported.
- **Solution:**
  1. Use the [describe-volumes](#) command to see how the volumes are exposed to the instance.
  2. Create a new launch configuration using the device name listed in the volume description.
  3. Update your Auto Scaling group with the new 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 configuration by specifying the device name in the `virtualName` parameter. For information about the device name format, see [Instance Store Device Names](#) in the *Amazon EC2 User Guide for Linux Instances*.
  2. Update your Auto Scaling group with the new 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 configuration are not supported on your instance.
- **Solution:**
  1. Create a new 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 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 configuration with the supported instance type.
  4. Update your Auto Scaling group with new placement group or launch configuration using the `update-auto-scaling-group` command.

## Troubleshooting 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 [Retrieving an Error Message \(p. 127\)](#).

When your EC2 instances fail to launch due to issues with your AMI, you might get one or more of the following error messages.

#### Error Messages

- The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 133)
- AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 133)
- Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed. (p. 133)
- Value (<ami ID>) for parameter virtualName is invalid. (p. 133)
- 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. 134)

## 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 configuration.
- **Solution:**
  1. Create a new launch configuration using a valid AMI.
  2. Update your Auto Scaling group with the new 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 configuration.

## Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed.

- **Cause:** The Linux AMI with hvm virtualization cannot be used to launch a non-cluster compute instance.
- **Solution:**
  1. Create a new launch configuration using an AMI with a virtualization type of paravirtual to launch a non-cluster compute instance.
  2. Update your Auto Scaling group with the new launch configuration using the [update-auto-scaling-group](#) command.

## Value (<ami ID>) for parameter virtualName is invalid.

- **Cause:** Incorrect value. The `virtualName` parameter refers to the virtual name associated with the device.
- **Solution:**

**Auto Scaling Developer Guide**  
**The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86\_64). Launching ec2 instance failed.**

---

1. Create a new 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 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 configuration does not match the image architecture.
- **Solution:**
  1. Create a new 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 configuration using the [update-auto-scaling-group](#) command.

## Troubleshooting 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 [Retrieving an Error Message](#) (p. 127).

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.

### Error Messages

- [Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.](#) (p. 134)
- [There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.](#) (p. 135)
- [EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.](#) (p. 135)
- [EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed.](#) (p. 135)
- [The security token included in the request is invalid. Validating load balancer configuration failed.](#) (p. 135)

## Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.

- **Cause 1:** The load balancer has been deleted.
- **Solution 1:**

1. Check to see if your load balancer still exists. You can use the [describe-load-balancers](#) command.
  2. If you see your load balancer listed in the response, see **Cause 2**.
  3. If you do not see your load balancer listed in the response, you can either create a new load balancer and then create a new Auto Scaling group or you can create a new Auto Scaling group without the load balancer.
- **Cause 2:** The load balancer name was not specified in the right order when creating the Auto Scaling group.
  - **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).

## **The security token included in the request is invalid. Validating load balancer configuration failed.**

- **Cause:** Your AWS account might have expired.
- **Solution:** Check if your AWS account is valid. Go to <http://aws.amazon.com> and click **Sign Up Now** to open a new account.

## Troubleshooting Auto Scaling: Capacity Limits

This page provides information about issues with the capacity limits of your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see [Retrieving an Error Message](#) (p. 127).

If your EC2 instances fail to launch due to issues with the capacity limits of your Auto Scaling group, you might get one or more of the following error messages.

### Error Messages

- [We currently do not have sufficient <instance type> capacity in the Availability Zone you requested \(<requested Availability Zone>\)... \(p. 136\)](#)
- [<number of instances> instance\(s\) are already running. Launching EC2 instance failed. \(p. 136\)](#)

### We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)...

- **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, Auto Scaling cannot support your instance type in your requested Availability Zone.
- **Solution:**
  1. Create a new launch configuration by following the recommendations in the error message.
  2. Update your Auto Scaling group with the new launch configuration using the [update-auto-scaling-group](#) command.

### <number of instances> instance(s) are already running. Launching EC2 instance failed.

- **Cause:** The Auto Scaling group has reached the limit set by the `DesiredCapacity` parameter.
- **Solution:**
  - Update your Auto Scaling group by providing a new value for the `--desired-capacity` parameter using the [update-auto-scaling-group](#) command.
  - If you've reached your limit for number of EC2 instances, you can request an increase. For more information, see [AWS Service Limits](#).

# Auto Scaling Command Line Interface (CLI)

---

You can use the Auto Scaling command line interface (CLI) to create and manage your Auto Scaling resources from the command line. For more information, download [as-cli.pdf](#), which is the final version of the Auto Scaling Developer Guide that covers the Auto Scaling CLI.

**Important**

We recommend that you use the AWS CLI instead of the Auto Scaling CLI. To get started, see the [AWS Command Line Interface User Guide](#). For more information about the AWS CLI commands for Auto Scaling, see [autoscaling](#) in the *AWS Command Line Interface Reference*.

# Auto Scaling Resources

---

The following related resources can help you as you work with this service.

- [Auto Scaling](#) – The primary web page for information about Auto Scaling.
- [Auto Scaling Technical FAQ](#) – The FAQ covers questions developers have asked about Auto Scaling.
- [Amazon EC2 Discussion Forum](#) – Get help from the community of developers.
  
- [AWS Developer Tools](#) – Links to developer tools and resources that provide documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
- [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 web page 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 Auto Scaling documentation.

Change	Description	Release Date
Support for ClassicLink	You can now 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 <a href="#">Linking EC2-Classic Instances to a VPC (p. 60)</a> .	19 January 2015
Support for life-cycle hooks	You can now hold your newly launched or terminating instances in a pending state while you perform actions on them. For more information, see <a href="#">Controlling How Instances Launch and Terminate (p. 62)</a> .	30 July 2014
Support for detaching instances from an Auto Scaling group	You can now detach instances from an Auto Scaling group. For more information, see <a href="#">Detach EC2 Instances From Your Auto Scaling Group (p. 98)</a> .	30 July 2014
Support for putting instances into a Standby state	You can now put instances that are in an <code>InService</code> state into a <code>Standby</code> state. For more information, see <a href="#">Auto Scaling InService State (p. 12)</a> .	30 July 2014
Support for managing Auto Scaling group tags using the AWS Management Console	You can now manage your Auto Scaling groups using the AWS Management Console. For more information, see <a href="#">Tagging Auto Scaling Groups and Instances (p. 73)</a> .	01 May 2014
Support for launching dedicated instances	You can now launch Dedicated Instances by specifying a placement tenancy attribute when you create a launch configuration. For more information, see <a href="#">Instance Placement Tenancy (p. 59)</a> .	23 April 2014



Change	Description	Release Date
Use an EC2 instance to create a new launch configuration or a new Auto Scaling group	You can now 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 <a href="#">Create a Launch Configuration Using an EC2 Instance (p. 50)</a> For information about creating an Auto Scaling group using an EC2 instance, see <a href="#">Create an Auto Scaling Group from an EC2 Instance (p. 56)</a> .	02 January 2014
Enable Auto Scaling for an existing EC2 instance	You can now enable Auto Scaling for an EC2 instance by attaching the instance to an existing Auto Scaling group. For more information, see <a href="#">Attach EC2 Instances to Your Auto Scaling Group (p. 95)</a> .	02 January 2014
Account limits	You can now find out the number of Auto Scaling resources allowed for your account. For more information, see <a href="#">Auto Scaling Limits (p. 14)</a> .	02 January 2014
Support for Auto Scaling in the AWS Management Console	You can now access Auto Scaling using the AWS Management Console. For more information, see <a href="#">Getting Started with Auto Scaling (p. 16)</a> .	10 December 2013
Assign public IP address to an instance launched into a VPC	You can now assign a public IP address to an instance launched into a VPC. For more information, see <a href="#">Auto Scaling and Amazon Virtual Private Cloud (p. 58)</a> .	19 September 2013
Content restructure	Created a new section for managing Auto Scaling groups and moved all the related procedures in this new section. For more information, see <a href="#">Configuring Your Auto Scaling Groups (p. 85)</a> .	11 June 2013
New tutorial	Added a tutorial for using Amazon SQS queues to establish thresholds that Auto Scaling can use to increase or decrease the capacity of your Auto Scaling group. For more information, see <a href="#">Scaling Based on Amazon SQS (p. 42)</a> .	12 March 2013
Content restructure	Changed the title of <i>Configuring Auto Scaling</i> to <i>Configure a Scaling Plan</i> . Added information about configuring a scaling plan for your Auto Scaling group. For more information, see <a href="#">Scaling the Size of Your Auto Scaling Group (p. 22)</a> .	08 February 2013
Instance termination policy	You can now specify the instance termination policy for Auto Scaling to use when terminating EC2 instances in your Auto Scaling group. For more information, see <a href="#">Choosing a Termination Policy for Your Auto Scaling Group (p. 26)</a> .	17 September 2012
Launching EC2 instances with an IAM role	You can now use Auto Scaling groups to launch EC2 instances with an IAM instance profile. You can use this feature to assign IAM roles to your instances, allowing applications that run on it to access other AWS services securely. For more information, see <a href="#">Launch Auto Scaling Instances with an IAM Role (p. 48)</a> .	11 June 2012

Change	Description	Release Date
Running Spot Instances	You can now run Spot Instances in Auto Scaling groups by specifying a Spot Instance bid price in your launch configuration. For more information, see <a href="#">Launching Spot Instances in Your Auto Scaling Group</a> (p. 76).	7 June 2012
Updated Troubleshooting Auto Scaling section	Added information about the issues, potential causes, and the steps you can take to resolve problems with Auto Scaling. For more information, see <a href="#">Troubleshooting Auto Scaling</a> (p. 127).	15 May 2012
Tagging Auto Scaling groups and EC2 instances	You can now tag Auto Scaling groups and specify that the tag also applies to EC2 instances launched after the tag was created. For more information, see <a href="#">Tagging Auto Scaling Groups and Instances</a> (p. 73).	26 January 2012
Amazon SNS integration	<p>Auto Scaling now supports Amazon SNS so that you can use it to receive notifications whenever Auto Scaling launches or terminates EC2 instances. For more information, see <a href="#">Getting Notifications When Your Auto Scaling Group Changes</a> (p. 119).</p> <p>Auto Scaling also added the following new features:</p> <ul style="list-style-type: none"> <li>• The ability to set up recurring scaling activities using cron syntax. For more information, see the <a href="#">PutScheduledUpdateGroupAction</a> API command.</li> <li>• 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 <a href="#">ProcessType</a> API data type.</li> <li>• The <code>ForceDelete</code> flag in the <code>DeleteAutoScalingGroup</code> command that tells 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 <a href="#">DeleteAutoScalingGroup</a> API command.</li> </ul>	20 July 2011
Updated example	Added a missing <i>Dimensions</i> parameter to examples that call the <code>PutMetricAlarm</code> command. For more information, see <a href="#">Load Balance Your Auto Scaling Group</a> (p. 86).	12 January 2011
Scheduled scaling actions	Added the ability to create scheduled scaling actions. For more information, see <a href="#">Scheduled Scaling</a> (p. 40).	2 December 2010
Amazon VPC support	Added support for Amazon VPC. For more information, see <a href="#">Auto Scaling and Amazon Virtual Private Cloud</a> (p. 58).	2 December 2010
HPC clusters	Added support for high performance computing (HPC) clusters.	2 December 2010

<b>Change</b>	<b>Description</b>	<b>Release Date</b>
Health check	Added the capability to use Elastic Load Balancing health check as the health check for Auto Scaling-managed EC2 instances. For more information, see <a href="#">Add an Elastic Load Balancing Health Check to your Auto Scaling Group (p. 91)</a> .	2 December 2010
CloudWatch alarms	Removed the older Trigger mechanism and redesigned Auto Scaling to use the CloudWatch alarm feature. For more information, see <a href="#">Dynamic Scaling (p. 32)</a> .	2 December 2010
Suspend and resume scaling	Added support for suspending and resuming scaling processes.	2 December 2010
IAM support	Auto Scaling now integrates with IAM. For more information, see <a href="#">Controlling Access to Your Auto Scaling Resources (p. 45)</a> .	2 December 2010

# AWS Glossary

---

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.