
Auto Scaling

Developer Guide

API Version 2011-01-01



Auto Scaling: Developer Guide

Copyright © 2014 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, Cloudfront, CloudTrail, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. 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

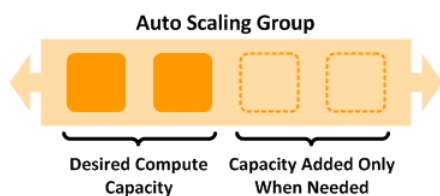
What is Auto Scaling?	1
Auto Scaling Benefits: An Example	1
Next Steps	3
How Auto Scaling Works	3
Groups, launch configurations, and scaling plans	4
Auto Scaling: An Example	5
Availability Zones and Regions	6
Limits	7
Auto Scaling Group Lifecycle	7
Auto Scaling Basic Lifecycle	7
Auto Scaling Instance States	9
Launch Configuration	12
Auto Scaling Group	12
Scaling Plans	12
Auto Scaling Policy	13
Setting Up	14
Sign Up	14
AWS Management Console	15
Command Line Interface	16
Setting the Java Home Variable	16
Setting Up the CLI	17
Verify if the Auto Scaling CLI is Installed	20
Query Request	23
Signing Query Requests	23
AWS SDKs	24
AWS Credentials	25
Log In with an Amazon Login and Password	25
View Your AWS Access Credentials	26
Get Your Access Key ID	26
Create an X.509 Certificate and Private Key	26
View Your Account ID	27
Getting Started	28
Getting Started Using the Console	28
Step 1: Create a Launch Configuration	29
Step 2: Create an Auto Scaling Group	31
Step 3: Verify Your Auto Scaling Group	33
Step 4: [Optional] Delete Your Auto Scaling Set Up	34
Getting Started Using the Command Line Interface	34
Using the CLI	35
Using the Query API	39
Planning your Auto Scaling Group	44
Scaling Your Group	45
Multiple scaling policies	46
Understanding cooldowns	46
Choosing a Termination Policy	49
Maintaining a Fixed Number of Running EC2 Instances	53
Manual Scaling	54
Dynamic Scaling	57
Controlling Access to Your Auto Scaling Resources	105
Use IAM with Auto Scaling	105
Launch Auto Scaling Instances with an IAM Role	108
Creating Launch Configurations	111
Create a Launch Configuration Using an EC2 Instance	111
Creating Your Auto Scaling Groups	122
Create an Auto Scaling Group Using an EC2 Instance ID	122

Auto Scaling in Amazon VPC	128
Controlling instances	137
Introducing lifecycle hooks	138
Lifecycle hook examples	143
Tagging instances	158
Tag Restrictions	159
Add or Modify Tags for Your Auto Scaling Group	159
Delete Tags	162
Launching Spot Instances in Your Auto Scaling Group	162
Launching Spot Instances in the AWS Management Console	163
Launching Spot Instances Using the CLI	172
Configuring Your Auto Scaling Groups	184
Load Balance Your Auto Scaling Group	185
Set Up a Scaled and Load-Balanced Application	186
Add an Elastic Load Balancing Health Check to your Auto Scaling Group	195
Expand Your Scaled and Load-Balanced Application to an Additional Availability Zone	197
Attach EC2 Instances to Your Auto Scaling Group	202
Attaching Instances Using the Command Line Interface	203
Attaching Instances Using the Query API	204
Detach EC2 Instances From Your Auto Scaling Group	208
Detaching Instances Using the Command Line Interface	208
Detaching Instances Using the Query API	209
Merging Auto Scaling Groups	212
Merge Zones Using the Command Line Interface	213
Merge Zones Using the Query API	215
Temporarily Removing Instances	216
Troubleshooting Instances	217
Upgrading or Modifying Instances	223
Suspend and Resume Process	230
Suspend Processes Using the Command Line Interface	231
Suspend Processes Using the Query API	232
Shut Down Your Auto Scaling Process	232
Shutting Down Using the Command Line Interface	233
Shutting Down Using the Query API	236
Monitoring Your Auto Scaling Instances	241
Amazon CloudWatch Alarms	241
Activating Detailed Instance Monitoring for Auto Scaling	242
Activating Basic Instance Monitoring for Auto Scaling	243
Auto Scaling Group Metrics	243
Auto Scaling Group Metrics Table	244
Dimensions for Auto Scaling Group Metrics	244
Health Checks	244
Instance Health Check	245
Getting Notifications When Your Auto Scaling Group Changes	248
Configure Amazon SNS	248
Configure your Auto Scaling Group to Send Notifications	249
Test the Notification Configuration	251
Verify That You Received Notification of the Scaling Event	253
Delete the Notification Configuration	254
Logging Auto Scaling API Calls By Using AWS CloudTrail	254
Auto Scaling Information in CloudTrail	255
Understanding Auto Scaling Log File Entries	255
Troubleshooting	258
Retrieving an Error Message	258
Instance Launch Failure	261
The security group <name of the security group> does not exist. Launching EC2 instance failed.	261

The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed.	261
The requested configuration is currently not supported.	261
AutoScalingGroup <Auto Scaling group name> not found.	262
The requested Availability Zone is no longer supported. Please retry your request	262
Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)....	262
You are not subscribed to this service. Please see http://aws.amazon.com	262
Invalid device name upload. Launching EC2 instance failed.	262
Value (<name associated with the instance storage device>) for parameter virtualName is invalid....	263
EBS block device mappings not supported for instance-store AMIs.	263
Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.	263
Amazon EC2 AMIs	264
The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.	264
AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.	264
Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed.	264
Value (<ami ID>) for parameter virtualName is invalid.	265
The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching ec2 instance failed.	265
Load Balancer Configuration	265
Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.	265
There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.	266
EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.	266
The security token included in the request is invalid. Validating load balancer configuration failed.	266
Capacity Limits	266
We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)....	267
<number of instances> instance(s) are already running. Launching EC2 instance failed.	267
Resources	268
Document History	269
AWS Glossary	273

What is Auto Scaling?

Auto Scaling is an AWS service that allows you to increase or decrease the number of EC2 instances within your application's architecture. With Auto Scaling, you create collections of EC2 instances, called Auto Scaling groups. You can create these groups from scratch, or from existing EC2 instances that are already in production.



You can create as many Auto Scaling groups as you need. For example, if an application consists of a web tier and an application tier, you can create two Auto Scaling groups—one for each tier. Each Auto Scaling group can contain one or more scaling policies—these policies define when Auto Scaling launches or terminated EC2 instances within the group.

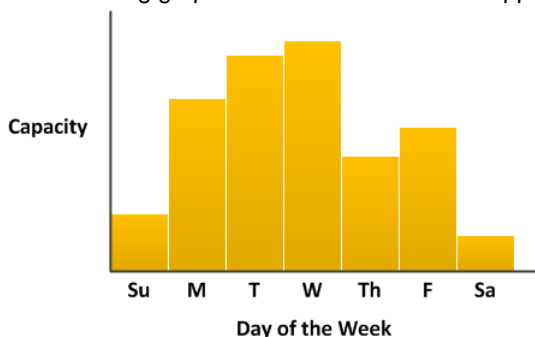
Adding Auto Scaling to your network architecture is one way to maximize the benefits of the AWS cloud. With Auto Scaling, you can make your applications:

- More fault tolerant. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch a new instance to replace it.
- More highly available. You can configure Auto Scaling to use multiple subnets or Availability Zones. If one subnet or Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
- Increase and decrease in capacity only when needed. Unlike on-premises solutions, with Auto Scaling you can have your network scale dynamically. You also don't pay for Auto Scaling. Instead, you pay only for the EC2 instances launched, and only for as long as you use them.

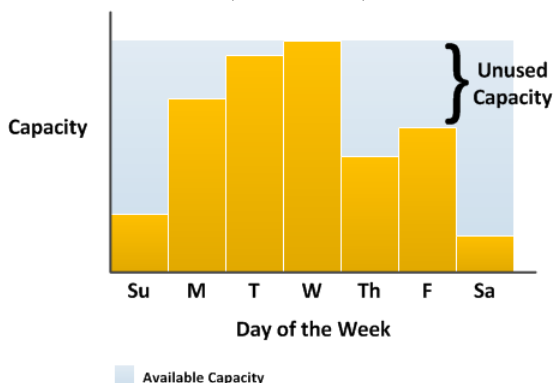
Auto Scaling Benefits: An Example

To better 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.

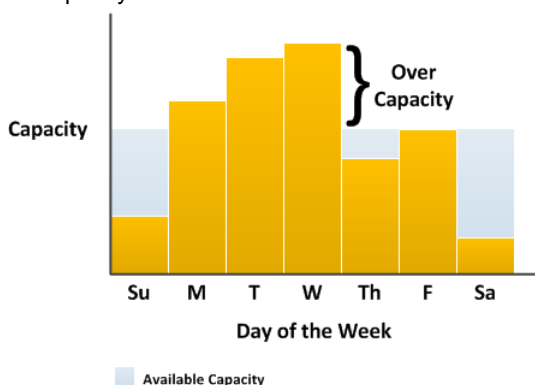
The following graph shows how much of the application's capacity is used over the course of a week.



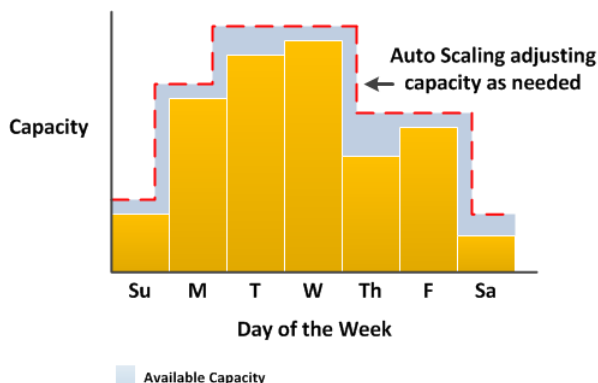
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity.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



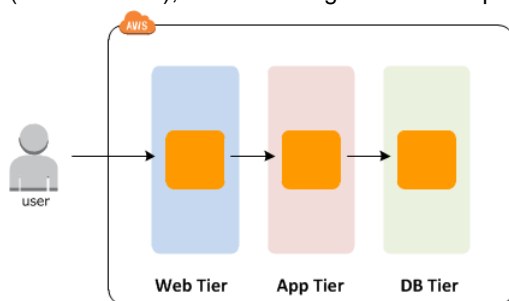
Next Steps

The remaining topics in this section provide a more detailed look at how Auto Scaling works. If you're new to Auto Scaling, we recommend that you review the sections [How Auto Scaling Works \(p. 3\)](#) and [Auto Scaling Group Lifecycle \(p. 7\)](#).

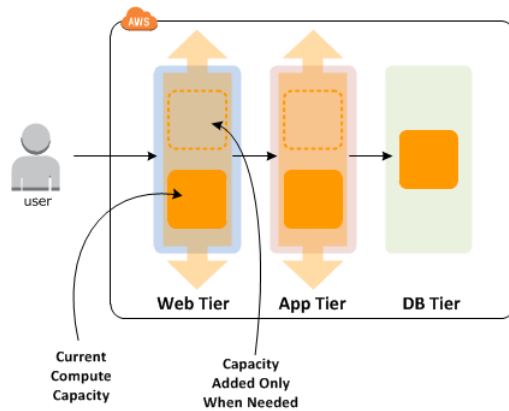
There are no additional fees with Auto Scaling, so it's easy to try and see how it can benefit your AWS architecture. To begin, review our [Getting Started with Auto Scaling \(p. 28\)](#) section to create a standalone Auto Scaling group and see how it responds when an instance in that group terminates. If you already have instances running in AWS, you can create an Auto Scaling group using an existing EC2 instance, and remove your instance from the group at any time. After you are familiar with how Auto Scaling works, review the topics [Planning your Auto Scaling Group \(p. 44\)](#) to learn how to make the most of Auto Scaling in your applications.

How Auto Scaling Works

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's more highly available and fault tolerant.



Note

This section assumes that you are familiar with Amazon Elastic Compute Cloud (Amazon EC2), and that you are using EC2 instances. For more information, see [Amazon EC2 Instances](#).

Topics

- [Groups, launch configurations, and scaling plans \(p. 4\)](#)
- [Auto Scaling: An Example \(p. 5\)](#)
- [Availability Zones and Regions \(p. 6\)](#)
- [Auto Scaling Limits \(p. 7\)](#)

Groups, launch configurations, and scaling plans

Groups, launch configurations, and scaling plans

	Groups When you use Auto Scaling, your EC2 instances are categorized into Auto Scaling <i>groups</i> for the purposes of instance scaling and management. You create Auto Scaling groups by defining the minimum, maximum, and, optionally, the desired number of running EC2 instances the group must have at any point in time.
	Launch Configurations Your Auto Scaling group uses a <i>launch configuration</i> to launch EC2 instances. You create the launch configuration by providing information about the image you want Auto Scaling to use to launch EC2 instances. The information can be the image ID, instance type, key pairs, security groups, and block device mapping. To learn more about Amazon machine images (AMI), see AMI Basics .

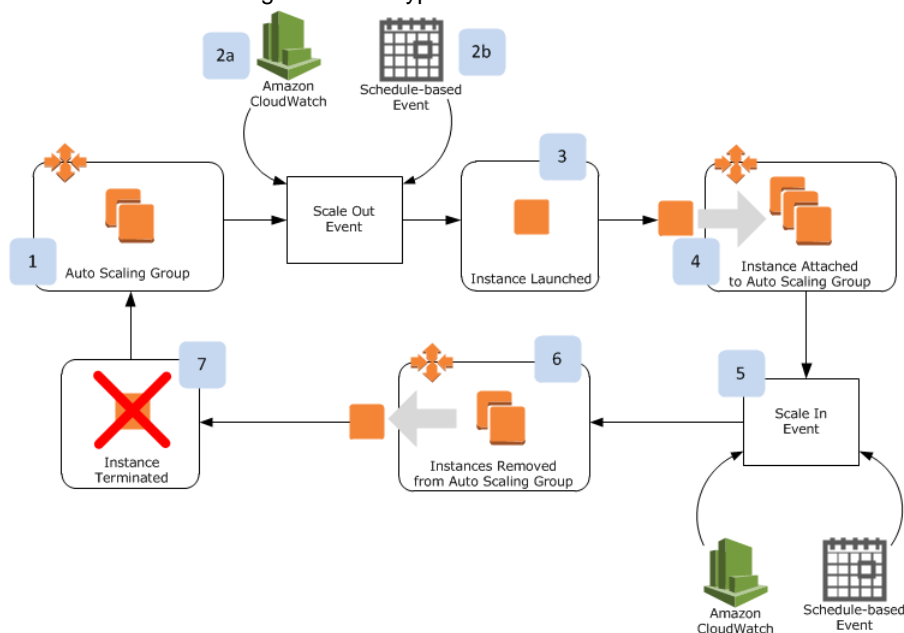


Scaling Plans

In addition to creating a launch configuration and an Auto Scaling group, you must also create a *scaling plan* for your Auto Scaling group. A scaling plan tells Auto Scaling when and how to scale. You can create a scaling plan based on the occurrence of specified conditions (dynamic scaling) or you can create a plan based on a specific schedule.

Auto Scaling: An Example

Here's how Auto Scaling works in a typical AWS environment.



1. We'll start with an Auto Scaling group set to have a desired capacity of two instances.
2. A scale out event occurs. This is an event that instructs Auto Scaling to launch an additional instance. A scale out event could be something like an CloudWatch alarm (item 2a in the diagram), or a schedule-based scaling policy (item 2b in the diagram) that launches instances at a specific day and time. See [Auto Scaling Basic Lifecycle \(p. 7\)](#) for additional examples.
3. Auto Scaling launches and configures the instance.
4. When the instance is fully configured, Auto Scaling attaches it to the Auto Scaling group.
5. Eventually, a corresponding scale in event occurs. A scale in event is like a scale out event, except that these types of events instruct Auto Scaling to terminate one or more instances. See [Auto Scaling Basic Lifecycle \(p. 7\)](#) for some examples of scale in events.
6. Auto Scaling removes the instances from the group and marks it for termination.
7. The instance terminates.

See [Auto Scaling Group Lifecycle \(p. 7\)](#) to learn more about how instances move through an Auto Scaling group, and what actions you can take during each state of an instance's lifecycle.

Availability Zones and Regions

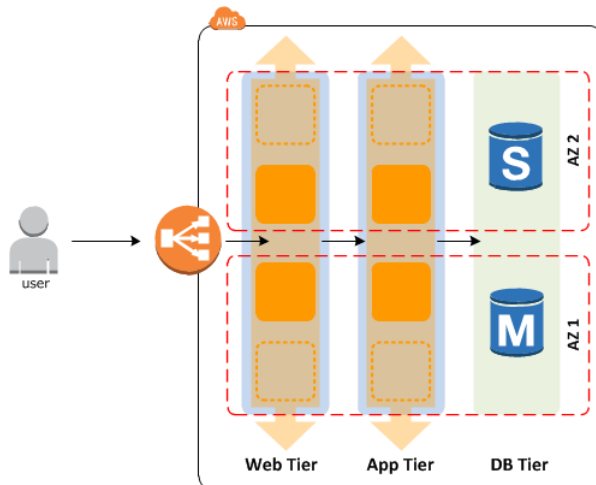
Amazon cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data centers are in several physical locations categorized by *regions* and *Availability Zones*. Regions are large and widely dispersed geographic locations. Availability Zones are distinct locations within a region 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 this product's regions and endpoints, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Auto Scaling lets you 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 that come from one or more EC2 *Availability Zones* within the same region. However, Auto Scaling group cannot span multiple regions.

Instance Distribution and Balance Across Multiple Zones

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 will attempt to launch in other zones until it succeeds.



Certain operations and conditions can cause your Auto Scaling group to become unbalanced between the 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.

Under all the above conditions, Auto Scaling launches new instances before attempting to terminate old ones, so a rebalancing activity will not compromise the performance or availability of your application.

Multi-Zone Instance Counts When Approaching Capacity

Because Auto Scaling always attempts to launch new instances before terminating old ones when attempting to balance across multiple zones, 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.

Auto Scaling Limits

Your AWS account comes with default limits on resources for Auto Scaling and other Amazon Web Services. Unless otherwise noted, each limit is per region. There is a default limit of 20 Auto Scaling groups and 100 launch configurations per region.

You can go to [AWS Service Limits](#) and select `Auto Scaling Limits` or any other service listed on the page to see its default limits.

If you reach the limit for the number of Auto Scaling groups or the number of launch configurations, you can go to [Support Center](#) and place a request to raise the limit.

You can see the number of Auto Scaling resources currently allowed for your AWS account either by using the `as-describe-account-limits` command or by calling the `DescribeAccountLimits` action.

For more information on installing and using the Auto Scaling command line interface (CLI) commands, the Query API actions or other interfaces, see [Setting Up Auto Scaling Interfaces](#) (p. 14)

Auto Scaling Group 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 Cycle 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.

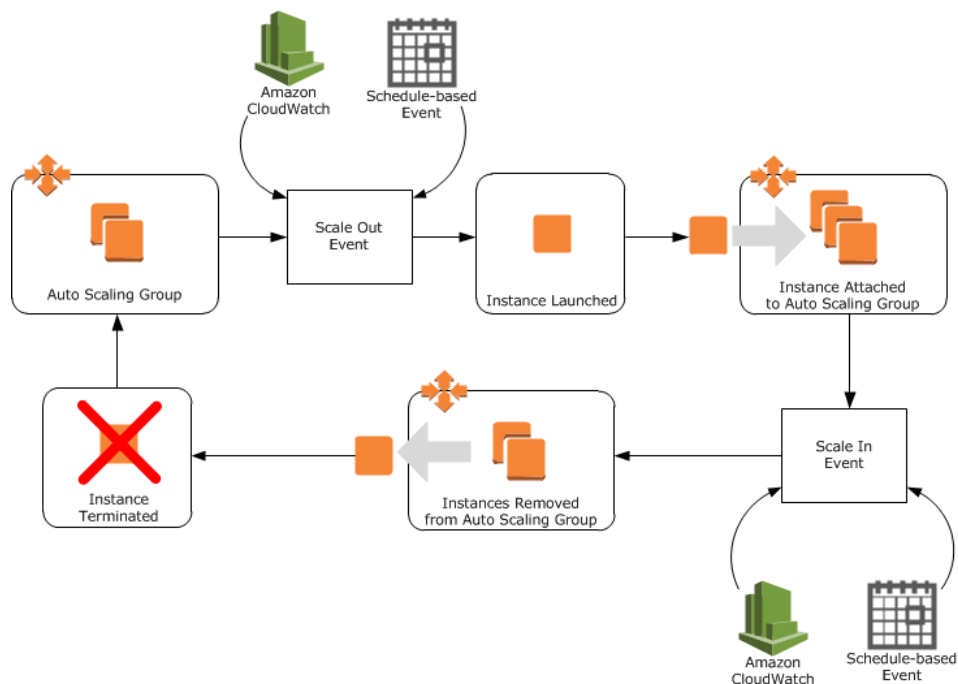
Topics

- [Auto Scaling Basic Lifecycle](#) (p. 7)
- [Auto Scaling Instance States](#) (p. 9)

Auto Scaling Basic Lifecycle

In this section, we describe the basic lifecycle of instances in an Auto Scaling group. This lifecycle applies to most implementations of Auto Scaling and is a great place to start if you are considering adding Auto Scaling to your application's architecture.

The following illustration shows the basic lifecycle of instances within an Auto Scaling group.



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

Scale out event

This event informs the Auto Scaling group to launch one or more new instances and add them to the application. Some examples of scale out events:

- You [manually](#) chose to increase the number of instances, either by setting a new minimum number of instances, or by configuring the desired capacity for the Auto Scaling group.
- You [created a Amazon CloudWatch alarm](#) to monitor your application.
- You [created a schedule-based policy](#) to scale out your application at a specific time.
- An existing instance fails a required number of health checks, or you [manually configure an instance](#) to have a have an `Unhealthy` status.

Instances launched

After a scale out event occurs, the Auto Scaling group uses its assigned launch configuration to launch one or more Amazon EC2 instances. The number of instances launched depends on how you configured your Auto Scaling group's [scaling policies](#). Instances that have launched but are not yet fully configured are typically in the [Pending \(p. 10\)](#) 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.

Important

Billing starts when the boot sequence of the instance is initiated, and ends when the instance terminates. For more information, see the [Billing section](#) of the Amazon EC2 FAQ.

Instance attached to the Auto Scaling group

When an instance has launched and is fully configured, it is put into service and attached to the Auto Scaling group load balancer. 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 event

It is important that each scale out event is matched to a corresponding scale in event. This helps ensure that your application's resources match the demand for those resources as closely as possible. As with scale out events, a scale in event can be one of a number of actions, including [manual configuration](#), a [CloudWatch alarm](#), a [schedule-based policy](#), or an instance failure.

Instance detached from an Auto Scaling group

When 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. 49). Instances that are in the process of detaching from the Auto Scaling group and shutting down are in the [Terminating](#) (p. 11) 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.

Instance terminated

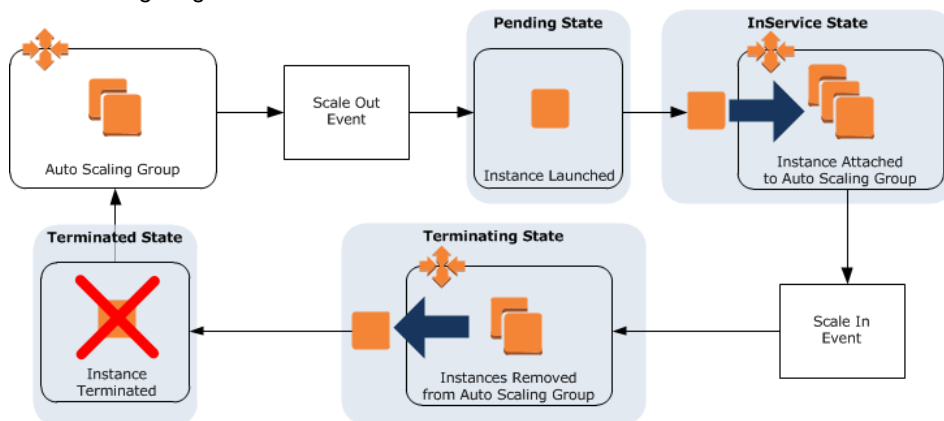
Finally, the instance is completely terminated.

Auto Scaling Instance States

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

- [Pending](#) (p. 10)
- [InService](#) (p. 10)
- [Terminating](#) (p. 11)
- 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:

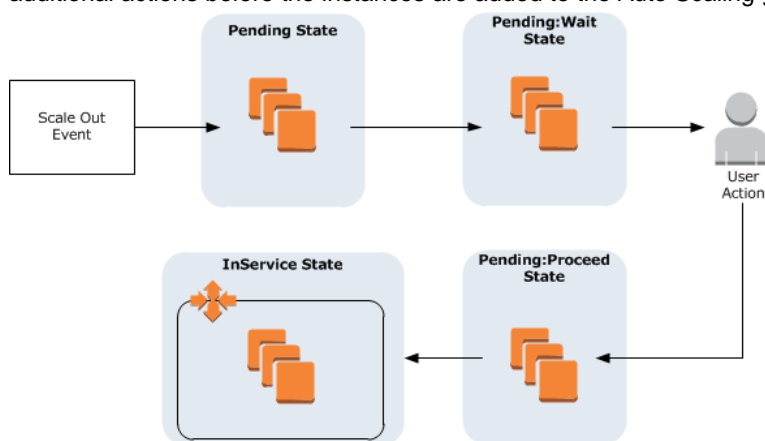
State	Action
Pending	Installing Software to Pending Instances (p. 143) Filling a Cache of Servers (p. 147)
InService	Updating or Modifying Instances in an Auto Scaling Group (p. 223) Troubleshooting Instances in an Auto Scaling Group (p. 217)

State	Action
Terminating	Analyzing an Instance Before Termination (p. 151) Retrieving Logs from Terminating Instances (p. 154)

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 \(p. 143\)](#)
- [Filling a Cache of Servers \(p. 147\)](#)

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.

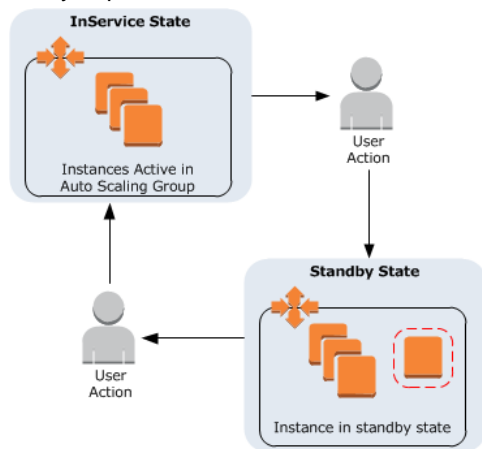
Auto Scaling InService State

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.



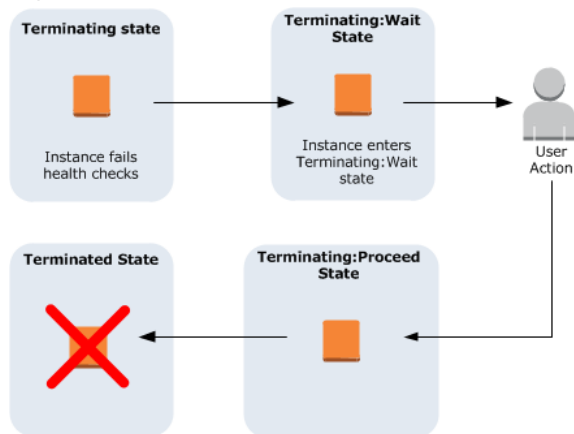
Examples of when you might put instances into the `Standby` state include:

- [To update or modify \(p. 223\)](#) the instance
- [To troubleshoot an instance \(p. 217\)](#) 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 \(p. 151\)](#) to understand why it failed
- [Retrieving logs \(p. 154\)](#) 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](#).

Launch Configuration

A *launch configuration* is a template that the Auto Scaling group uses to launch EC2 instances. 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. When you create your Auto Scaling group, you must associate it with a launch configuration. You can attach only one launch configuration to an Auto Scaling group at a time. Launch configurations cannot be modified. They are immutable. If you want to change the launch configuration of your Auto Scaling group, you must first create a new launch configuration and then update your Auto Scaling group by attaching the new launch configuration. When you attach a new launch configuration to your Auto Scaling group, any new instances are launched using the new configuration parameters but existing instances are not affected. For information about creating launch configuration, see [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#).

You can have a maximum of 100 launch configurations for your AWS account. If you reach the limit for number of launch configurations, go to [Support Center](#) and place a request to raise your launch configuration limit.

Auto Scaling Group

An *Auto Scaling group* is a representation of multiple EC2 instances that share similar characteristics, and that 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 or decrease the number of instances in that group to improve the performance of the application. You can use the Auto Scaling group to scale the number of instances automatically or maintain a fixed number of instances. You create Auto Scaling groups by defining the minimum, maximum, and desired number of running EC2 instances that the group must have at any given point of time.

An Auto Scaling group starts by launching the minimum or specified number of EC2 instances and then increases or decreases the number of running EC2 instances automatically according to conditions that you define. Auto Scaling also maintains the current instance levels by conducting periodic health checks on all the instances within the Auto Scaling group. If an EC2 instance within the Auto Scaling group becomes unhealthy, Auto Scaling terminates the unhealthy instance and launches a new one to replace it. This automatic scaling and maintenance of the instance levels in an Auto Scaling group is the core value of the Auto Scaling service. For information about creating an Auto Scaling group, see [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#).

Auto Scaling and Amazon EC2 provide default limits on the resources for an AWS account. For example, your AWS account comes with a default limit of 20 Auto Scaling groups. For more information about the limits for Auto Scaling and how to raise them, see [Auto Scaling Limits \(p. 7\)](#). For information about the limits for Amazon EC2, see [Amazon EC2 Limits](#) in the *AWS General Reference*.

Scaling Plans

Auto Scaling provides you with several ways to configure 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 Running EC2 Instances](#) (p. 53).

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. 54).

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. 78).

Scale based on demand

A more advanced way to scale your resources, scaling by policy, lets you define parameters that inform 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. 57).

Auto Scaling Policy

An *Auto Scaling policy* is a set of instructions for Auto Scaling on how to scale in (terminate EC2 instances) or scale out (launch EC2 instances) the Auto Scaling group. You can use Auto Scaling policies to initiate a launch instance or terminate instance activity for the Auto Scaling group. Use the [PutScalingPolicy](#) action or `as-put-scaling-policy` command to create Auto Scaling policies.

Auto Scaling policies can be used either to initiate a scaling activity manually (scale in or scale out) or based on a metric, such as traffic to your instances.

To initiate a scaling activity based on the traffic or any other metric, associate the Auto Scaling policy with a [Amazon CloudWatch Alarms](#) (p. 241) action. You can configure a CloudWatch alarm to monitor a single metric, such as average CPU usage of the EC2 instances. When the metric breaches a specified range, it invokes actions. If associated, the action then triggers an Auto Scaling policy. For information about using an Auto Scaling policy to initiate scaling activity based on a metric, see [Dynamic Scaling](#) (p. 57).

To use an Auto Scaling policy to initiate manual scaling, use the [ExecutePolicy](#) action or `as-execute-policy` command.

Setting Up Auto Scaling Interfaces

To use Auto Scaling, you'll need to sign up for an AWS account. Signing up allows you to access Auto Scaling and other services in AWS that you may need, such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon CloudWatch, Amazon Simple Notification Service (Amazon SNS), and Elastic Load Balancing.

After signing up, you need to install and configure the Auto Scaling interface you want to use to create, access, and manage your Auto Scaling resources.

You'll also need AWS credentials to access the AWS CLI, Auto Scaling CLI, the Auto Scaling Query API, and the Auto Scaling SDK.

Topics

- [Sign Up for AWS](#) (p. 14)
- [Sign in to Auto Scaling Using the AWS Management Console](#) (p. 15)
- [Install the Auto Scaling CLI](#) (p. 16)
- [Use Query Requests to Call Auto Scaling APIs](#) (p. 23)
- [Use the AWS SDKs](#) (p. 24)
- [Manage Your AWS Credentials](#) (p. 25)

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 EC2 instances.

If you already have an AWS account, skip this step. If you don't have an AWS account, use the following procedure to create one.

To sign up for an AWS account

1. Go to <http://aws.amazon.com> and click the **Sign Up** button.
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.

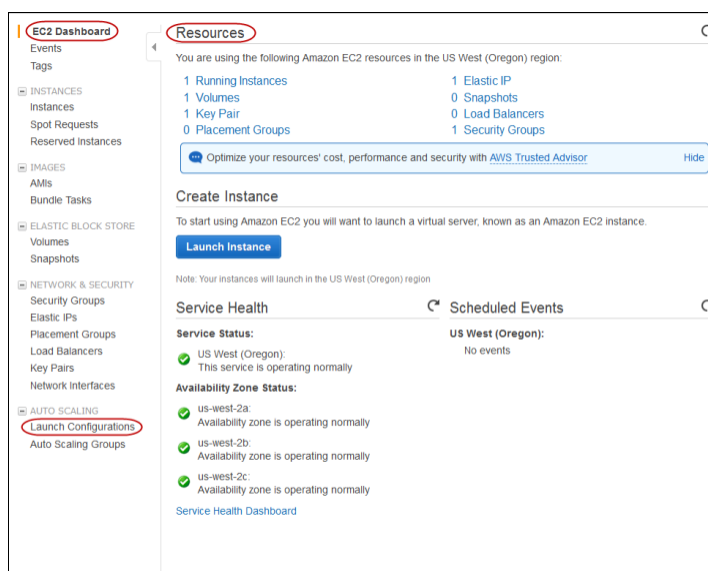
Sign in to Auto Scaling Using the AWS Management Console

The AWS Management Console is a point-and-click web-based interface you can use to access Auto Scaling and other AWS products. You can use the console to make requests to Auto Scaling and other AWS APIs.

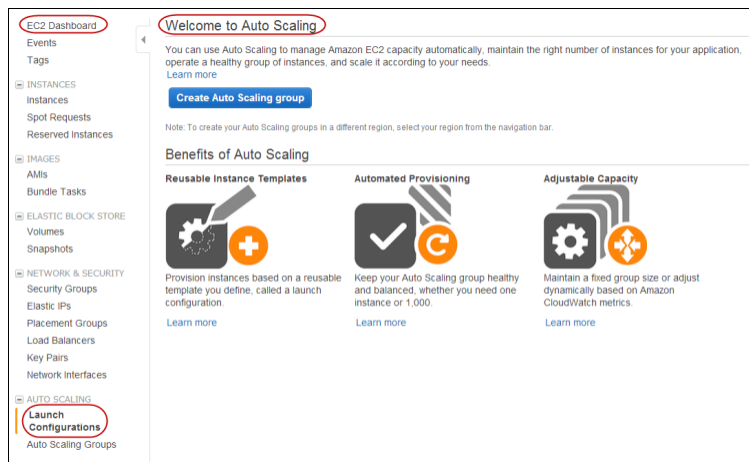
Auto Scaling resources can be accessed using the Amazon EC2 console in the AWS Management Console.

To sign in to the Auto Scaling using the Amazon EC2 console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 console **Resources** page, in the left navigation pane, under **Auto Scaling**, click **Launch Configurations** to start the Auto Scaling wizard.



3. If this is the first time you are using Amazon EC2 console to access Auto Scaling, your Auto Scaling wizard might look something like the following:



4. The Auto Scaling resources that you create are tied to a region that you specify and are not replicated across regions. For more information, see [Availability Zones and Regions \(p. 6\)](#).

The AWS Management Console selects a region for you by default. The default region is displayed in the navigation bar. If necessary, change the region. From the navigation bar, select the region that meets your needs.

Install the Auto Scaling CLI

Auto Scaling provides a command line interface (CLI) to access Auto Scaling functionality without using the AWS Management Console, the APIs, or the SDKs. The CLI wraps the API actions to provide multi-function commands. The CLI commands are written in Java and include shell scripts for both Windows and Linux/Unix/Mac OS X. The shell scripts are available as a self-contained ZIP file.

This section describes how to set up the Auto Scaling CLI.

Process for Installing the CLI

Setting the Java Home Variable (p. 16)
Setting Up the CLI (p. 17)
Verify if the Auto Scaling CLI is Installed (p. 20)

Note

As a convention, all command line text is prefixed with a generic **PROMPT>** command line prompt. The actual command line prompt on your computer is likely to be different. We also use **\$** to indicate a Linux/UNIX-specific command and **C:\>** for a Windows-specific command. Although we don't provide explicit instructions, the tools also work correctly on Mac OS X (which resemble the Linux and UNIX commands). The example output resulting from the command is shown immediately thereafter without any prefix.

Setting the Java Home Variable

The Auto Scaling command line toolCLIs read an environment variable (**JAVA_HOME**) on your computer to locate the Java runtime. The CLI requires Java version 5 or later to run. Either a JRE or JDK installation is acceptable.

To set the **JAVA_HOME** environment variable

1. If you do not have Java 1.5 or later installed, download and install it now. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to <http://java.oracle.com/>.
2. Set **JAVA_HOME** to the full path of the directory that contains a subdirectory named **bin** that in turn contains the Java executable. For example, if your Java executable is in the **/usr/jdk/bin** directory, set **JAVA_HOME** to **/usr/jdk**. If your Java executable is in **C:\jdk\bin**, set **JAVA_HOME** to **C:\jdk**.

Note

If you are using Cygwin, you must use Linux/UNIX paths (e.g., **/usr/bin** instead of **C:\usr\bin**) for **AWS_AUTO_SCALING_HOME** and **AWS_CREDENTIAL_FILE**. However, **JAVA_HOME** should have a Windows path. Additionally, the value cannot contain any spaces, even if the value is quoted or the spaces are escaped.

The following Linux/UNIX example sets **JAVA_HOME** for a Java executable in the **/usr/local/jre/bin** directory.

```
$ export JAVA_HOME=/usr/local/jre
```

The following Windows example uses **set** and **setx** to set `JAVA_HOME` for a Java executable in the `C:\java\jdk1.6.0_6\bin` directory. The **set** command defines `JAVA_HOME` for the current session and **setx** makes the change permanent.

```
C:\> set JAVA_HOME=C:\java\jdk1.6.0_6  
C:\> setx JAVA_HOME C:\java\jdk1.6.0_6
```

Note

- The **setx** command does not use the = sign.
- Don't include the `bin` directory in `JAVA_HOME`; that's a common mistake some users make. The CLI won't work if you do.
- The value for `JAVA_HOME` cannot contain any spaces, even if the value is quoted or the spaces are escaped. If the value contains a space character, the CLI returns an error when you add `JAVA_HOME` to your path in the next step.

3. Add your Java directory to your path before other versions of Java.

On Linux and UNIX, you can update your `PATH` as follows:

```
$ export PATH=$JAVA_HOME/bin:$PATH
```

On Windows, the syntax is slightly different:

```
C:\> set PATH=%JAVA_HOME%\bin;%PATH%  
C:\> setx PATH %JAVA_HOME%\bin;%PATH%
```

4. On Linux or UNIX, verify your `JAVA_HOME` setting with the command `$JAVA_HOME/bin/java -version`.

```
$ $JAVA_HOME/bin/java -version  
java version "1.5.0_09"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)  
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

The syntax is different on Windows, but the output is similar.

```
C:\> %JAVA_HOME%\bin\java -version  
java version "1.5.0_09"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)  
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

Setting Up the CLI

Topics

- [Get the CLI \(p. 18\)](#)

- [Set the Environment Variable for the CLI \(p. 18\)](#)
- [Manage Access for the CLI \(p. 19\)](#)
- [How to Change the Region \(p. 19\)](#)

To use the Auto Scaling CLI, you need to download it and set it up to use your AWS account.

Get the CLI

The CLI is available as a ZIP file in the [Auto Scaling Command Line Tools](#) website. These CLI tools are written in Java and include shell scripts for both Windows and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it.

Some additional setup is required for the CLI to use your AWS credentials. These are discussed next.

Set the Environment Variable for the CLI

The CLI depends on an environment variable (`AWS_AUTO_SCALING_HOME`) to locate supporting libraries. You need to set this environment variable before you can use the CLI. You should set this variable to the path of the directory into which the CLI was unzipped. This directory is named `AutoScaling-a.b.c.d` (a, b, c, and d are version/release numbers) and contains sub-directories named `bin` and `lib`.

The following Linux/UNIX example sets `AWS_AUTO_SCALING_HOME` for a directory named `as-1.0.12.0` in the `/usr/local` directory.

```
$ export AWS_AUTO_SCALING_HOME=/usr/local/as-1.0.12.0
```

The following Windows example sets `AWS_AUTO_SCALING_HOME` for a directory named `as-1.0.12.0` in the `C:\CLIs` directory.

```
C:\> set AWS_AUTO_SCALING_HOME=C:\CLIs\as-1.0.12.0
C:\> setx AWS_AUTO_SCALING_HOME C:\CLIs\as-1.0.12.0
```

In addition, you may want to add the CLI `bin` directory to your system `PATH` to avoid having to reference this directory specifically in every command. The examples in this guide assume that you have modified your `PATH` as noted.

On Linux and UNIX, you can update your `PATH` as follows:

```
$ export PATH=$PATH:$AWS_AUTO_SCALING_HOME/bin
```

On Windows, the syntax is slightly different:

```
C:\> set PATH=%PATH%;%AWS_AUTO_SCALING_HOME%\bin
C:\> setx PATH %PATH%;%AWS_AUTO_SCALING_HOME%\bin
```

Note

The Windows environment variables are reset when you close the command window. You might want to set them permanently with the `setx` command.

Consult the documentation for your version of Windows for more information.

Manage Access for the CLI

After you sign up for AWS, you must create access keys for the account. Your access keys consists of an access key ID and a secret access key. You must provide your access keys to the CLI to authenticate the commands that you issue from your account. The CLI reads your access keys from a credential file that you create on your local system.

You can either specify your credential file with the `--aws-credential-file` parameter every time you issue a command, or you can create an environment variable that points to the credential file on your local system. If the environment variable is properly configured, you can omit the `--aws-credential-file` parameter when you issue a command. The following procedure describes how to create a credential file and a corresponding `AWS_CREDENTIAL_FILE` environment variable.

To create a credential file on your local system

1. Retrieve your access keys

Although you can retrieve the access key ID from the **Security Credentials** page, you cannot retrieve the secret access key. You'll need to create new access keys if the secret access key was lost or forgotten. You can create new access keys for the account by going to the [Security Credentials](#) page. In the **Access Keys** section, click **Create New Root Key**.

2. Write down your secret key and access key ID, or save them.
3. Add your access key ID and secret access key to the file named `credential-file-path.template`:
 - a. Open the file `credential-file-path.template` included in your CLI archive.
 - b. Copy and paste your access key ID and secret access key into the file.
 - c. Rename the file and save it to a convenient location on your computer.
 - d. If you are using Linux, set the file permissions as follows:

```
$ chmod 600 credential-file-name
```

4. Set the `AWS_CREDENTIAL_FILE` environment variable to the fully qualified path of the file you just created.

The following Linux/UNIX example sets `AWS_CREDENTIAL_FILE` for `myCredentialFile` in the `/usr/local` directory.

```
$ export AWS_CREDENTIAL_FILE=/usr/local/myCredentialFile
```

The following Windows example sets `AWS_CREDENTIAL_FILE` for `myCredentialFile.txt` in the `C:\aws` directory.

```
C:\> set AWS_CREDENTIAL_FILE=C:\aws\myCredentialFile.txt  
C:\> setx AWS_CREDENTIAL_FILE C:\aws\myCredentialFile.txt
```

How to Change the Region

By default, the Auto ScalingCLI uses the US East (N. Virginia) region (`us-east-1`) with the `autoscaling.us-east-1.amazonaws.com` service endpoint URL. If your instances are in a different region, you

must specify the region where your instances reside. For example, if your instances are in Europe, you must specify the EU (Ireland) region by using the `--region eu-west-1` parameter or by setting the `AWS_AUTO_SCALING_URL` environment variable.

This section describes how to specify a different region by changing the service endpoint URL.

To specify a different region

1. View available regions in the [Regions and Endpoints](#) topic in the *AWS General Reference*.
2. If you want to change the service endpoint, set the `AWS_AUTO_SCALING_URL` environment variable as follows:

Note

Keep in mind that if you set the `EC2_REGION` environment variable, such as **us-east-1**, its value supersedes any value you set using `AWS_AUTO_SCALING_URL`.

- The following Linux/UNIX example sets `AWS_AUTO_SCALING_URL` to the EU (Ireland) region.

```
$ export AWS_AUTO_SCALING_URL=https://autoscaling.eu-west-1.amazonaws.com
```

- The following Windows example sets `AWS_AUTO_SCALING_URL` to the EU (Ireland) region.

```
C:\> set AWS_AUTO_SCALING_URL=https://autoscaling.eu-west-1.amazonaws.com  
  
C:\> setx AWS_AUTO_SCALING_URL https://autoscaling.eu-west-1.amazonaws.com
```

Verify if the Auto Scaling CLI is Installed

Before you begin using the CLI, you should verify that it is properly installed.

To verify your Auto Scaling installation and configuration

1. On your Linux or Windows workstation, open a new command prompt.
2. Type the command `as -cmd`.
3. You should see output similar to the following:

Command Name	Description
-----	-----
as-create-auto-scaling-group	Create a new Auto Scaling group.
as-create-launch-config	Creates a new launch configuration.
as-create-or-update-tags	Create or update tags.
as-delete-auto-scaling-group	Deletes the specified Auto
Scaling group.	
as-delete-launch-config	Deletes the specified launch
configuration.	
as-delete-notification-configuration	Deletes the specified notification
configuration.	
as-delete-policy	Deletes the specified policy.

as-delete-scheduled-action action.	Deletes the specified scheduled
as-delete-tags	Delete the specified tags
as-describe-adjustment-types types.	Describes all policy adjustment
as-describe-auto-scaling-groups Scaling groups.	Describes the specified Auto
as-describe-auto-scaling-instances Scaling instances.	Describes the specified Auto
as-describe-auto-scaling-notification-types notification types.	Describes all Auto Scaling noti
as-describe-launch-configs configurations.	Describes the specified launch
as-describe-metric-collection-types metric granularity types.	Describes all metric colle...
as-describe-notification-configurations Auto Scaling groups.	Describes all notification...given
as-describe-policies	Describes the specified policies.
as-describe-process-types types.	Describes all Auto Scaling process
as-describe-scaling-activities belonging to a group.	Describes a set of activities
as-describe-scheduled-actions actions.	Describes the specified scheduled
as-describe-tags	Describes tags
as-describe-termination-policy-types mination policy types.	Describes all Auto Scaling ter
as-disable-metrics-collection Scaling group metrics.	Disables collection of Auto
as-enable-metrics-collection Scaling group metrics.	Enables collection of Auto
as-execute-policy	Executes the specified policy.
as-put-notification-configuration the Auto Scaling group.	Creates or replaces notifi...or
as-put-scaling-policy Scaling policy.	Creates or updates an Auto
as-put-scheduled-update-group-action update group action.	Creates or updates a scheduled
as-resume-processes given Auto Scaling group.	Resumes all suspended Auto...
as-set-desired-capacity the Auto Scaling group.	Sets the desired capacity of
as-set-instance-health	Sets the health of the instance.
as-suspend-processes given Auto Scaling group.	Suspends all Auto Scaling ...
as-terminate-instance-in-auto-scaling-group	Terminates a given instance.
as-update-auto-scaling-group Scaling group.	Updates the specified Auto
help	
version	Prints the version of the CLI
tool and the API.	

For help on a specific command, type '*commandname* --help'

This completes your installation and configuration of the Auto Scaling command line tools. You're ready to start accessing Auto Scaling using the command line interface (CLI). For descriptions of all the Auto Scaling commands, see [Auto Scaling Quick Reference Card](#).

Use Query Requests to Call Auto Scaling APIs

Auto Scaling provides APIs that you can call by submitting a Query Request. Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named *Action* or *Operation* that specifies the API you are calling. Action is used throughout this documentation, although Operation is also supported for backward compatibility with other Amazon Web Services (AWS) Query APIs.

Calling the API using a Query request is the most direct way to access the web service, but requires that your application handle low-level details such as generating the hash to sign the request, and error handling. The benefit of calling the service using a Query request is that you are assured of having access to the complete functionality of the API.

Note

The Query interface used by AWS is similar to REST, but does not adhere completely to the REST principles.

Signing Query Requests

The signature format that AWS uses has been refined over time to increase security and ease of use. Auto Scaling supports Signature Version 2 and Signature Version 4. If you are creating new applications that use Auto Scaling, then we recommend using Signature Version 4 for signing your query requests. For more information, see the following topics in the *AWS General Reference*

- [Signature Version 4 Signing Process](#)
- [Signature Version 2 Signing Process](#)

Use the AWS SDKs

The AWS SDKs provide functions that wrap an API and take care of many of the connection details, such as calculating signatures, handling request retries, and error handling. The SDKs also contain sample code, tutorials, and other resources to help you get started writing applications that call AWS. Calling the wrapper functions in an SDK can greatly simplify the process of writing an AWS application. You can access Auto Scaling programmatically using the SDKs in Java, .NET, PHP, or Ruby.

A disadvantage of using the SDKs is that the implementation of the wrapper functions sometimes lags behind changes to the web service's API, meaning that there may be a period between the time that a new web service API is released and when a wrapper function for it becomes available in the SDKs. You can overcome this disadvantage by using the SDKs to generate a raw Query request.

The following table lists the available SDKs and third-party libraries you can use to access Auto Scaling programmatically.

Access Type	Description
AWS SDKs	<p>AWS provides the following SDKs:</p> <ul style="list-style-type: none">• AWS SDK for Java Documentation• AWS SDK for .NET Documentation• AWS SDK for PHP Documentation• AWS SDK for Ruby Documentation
Third-party libraries	<p>Developers in the AWS developer community also provide their own libraries, which you can find at the following AWS developer centers:</p> <ul style="list-style-type: none">• AWS Java Developer Center• AWS PHP Developer Center• AWS Python Developer Center• AWS Ruby Developer Center• AWS Windows and .NET Developer Center

Manage Your AWS Credentials

Topics

- [Log In with an Amazon Login and Password](#) (p. 25)
- [View Your AWS Access Credentials](#) (p. 26)
- [Get Your Access Key ID](#) (p. 26)
- [Create an X.509 Certificate and Private Key](#) (p. 26)
- [View Your Account ID](#) (p. 27)

This section describes how to manage the following Auto Scaling credentials:

- **Amazon login and password**—Used to sign up for Amazon EC2 and other services, view your bills, perform account-based tasks, and get many of your security credentials. Additionally, they are used by the AWS Management Console. For information, see [Log In with an Amazon Login and Password](#) (p. 25).
- **Access key ID** —Used to make Query and REST-based requests. Also commonly used by UI-based tools, such as ElasticFox. For more information, see [Get Your Access Key ID](#) (p. 26).
- **X.509 certificate and private key**—Used to make SOAP API requests. For more information, see [Create an X.509 Certificate and Private Key](#) (p. 26).
- **Account ID**—Used to share resources with other AWS accounts. For more information, see [View Your Account ID](#) (p. 27).

Log In with an Amazon Login and Password

Topics

The Amazon login and password enable you to sign up for services, view your bills, perform account-based tasks, and get many of your security credentials. You also use the login and password to perform Amazon EC2 tasks through the AWS Management Console.

To log in with your existing login and password

1. Go to the [AWS website](#).
2. Select an option from the **Your Account** menu.
The **Amazon Web Services Sign In** page appears.
3. Enter your e-mail address, select **I am a returning user and my password is**, enter your password, and click **Sign In**.

To get a new Amazon login and password

1. Go to the [AWS website](#).
2. Click **Create an AWS Account**.
The **Amazon Web Services Sign In** page appears.
3. Enter your e-mail address, select **I am a new user**, and click **Sign In**.
4. Follow the on-screen prompts to create a new account.

Note

It is important to keep your Amazon login and password secret as they can be used to view and create new credentials. As an increased security measure, Amazon offers Multi-Factor Authentication, which uses the combination of a physical device and passcode to log in to your AWS account. For more information, see <http://aws.amazon.com/mfa>.

View Your AWS Access Credentials

You can view your active AWS access credentials anytime using the AWS Management Console.

To view your AWS access credentials

1. Go to the [AWS web site](#).
2. Point to **Your Account** and select **Security Credentials**.

If you are not already logged in, you are prompted to do so.

3. All your access credentials associated with your AWS account are displayed on **Your Security Credentials** page. Move the mouse over each credential to see the description and click to see the details.

Get Your Access Key ID

When you create your AWS account, you have to create your access keys. Your access keys consist of an access key ID and a secret access key. The access keys are the most commonly used AWS credentials. You can use them to make Query and REST-based requests and to use the command line tools. They are also commonly used by UI-based tools, such as ElasticFox. You can use up to two sets of access keys at a time. You can generate new keys at any time or disable existing keys.

Important

To ensure the security of your AWS account, your access keys are accessible only during key and user creation. You must save the keys (for example, in a text file) if you want to be able to access it again. If your access keys are lost, you can create new keys.

To get your access key ID

1. Go to the [AWS web site](#).
2. Point to **Your Account** and select **Security Credentials**.

If you are not already logged in, you are prompted to do so.

3. In the **Your Security Credentials** page, select **Access Keys**.
4. The **Access Keys** pane opens to display the details of your access keys.
5. To disable an access key, click **Make Inactive**. To re-enable the key, click **Make Active**. Click **Delete** to delete the key.
6. If no access keys appear in the list, click **Create Access Key** and follow the on-screen prompts.

Create an X.509 Certificate and Private Key

The X.509 certificate and private key are used by the command line tools and SOAP. You can download the private key file one time. If you lose it, you need to create a new certificate. Up to two certificates can be active at any time.

To create a certificate

1. Go to the [AWS web site](#).

2. Point to **Your Account** and select **Security Credentials**.

If you are not already logged in, you are prompted to do so.

3. Click the **X.509 Certificates** tab.
4. Click **Create a New Certificate** and follow the on-screen prompts.

The new certificate is created and appears in the X.509 certificates list. You are prompted to download the certificate and private key files.

5. Create an .as directory (the "as" stands for "Auto Scaling") in your home directory, and save these files to it with the file names offered by your browser.

You should end up with a PEM-encoded X.509 certificate and a private key file.

View Your Account ID

The account ID identifies your account to AWS and enables other accounts to access resources that you want to share, such as Amazon EC2 AMIs and Amazon EBS snapshots.

To view your account ID

1. Go to the [AWS web site](#).
2. Point to **Your Account** and select **Security Credentials**.

If you are not already logged in, you are prompted to do so.

3. Scroll down to the **Account Identifiers** section.
4. Locate your AWS account ID.

Note

The account ID number is not a secret. When granting access to resources, make sure to specify the account ID without hyphens.

Getting Started with Auto Scaling

For any application in which you plan to use Auto Scaling, you must use certain building blocks to get started. This section walks you through the process for setting up the basic infrastructure that gets Auto Scaling started for your EC2 instances in the EC2-Classic platform using the AWS Management Console or the Auto Scaling command line interface (CLI).

If you want to create your basic Auto Scaling infrastructure within EC2-VPC , see [Auto Scaling in Amazon Virtual Private Cloud \(p. 128\)](#).

When you launch your instances using Auto Scaling, you secure it by specifying a key pair and security group. (This tutorial assumes that you are familiar with launching EC2 instances and have already created a key pair and a security group to use for the instances launched by Auto Scaling. For more information, see [Get Set Up for Amazon EC2](#)) in the *Amazon EC2 User Guide for Linux Instances* .

Before you proceed, make sure that you have completed the tasks described in [Setting Up Auto Scaling Interfaces \(p. 14\)](#) for using the interface of your choice to access Auto Scaling.

- [Getting Started with Auto Scaling Using the Console \(p. 28\)](#)
- [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#)

Getting Started with Auto Scaling Using the Console

The following step-by-step instructions will help you create a template that defines your instance, create an Auto Scaling group to maintain the healthy number of instances at all times, and optionally delete this basic Auto Scaling set up using the AWS Management Console, a point-and-click web-based interface.

Topics

- [Step 1: Create a Launch Configuration \(p. 29\)](#)
- [Step 2: Create an Auto Scaling Group \(p. 31\)](#)
- [Step 3: Verify Your Auto Scaling Group \(p. 33\)](#)
- [Step 4: \[Optional\] Delete Your Auto Scaling Set Up \(p. 34\)](#)

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.

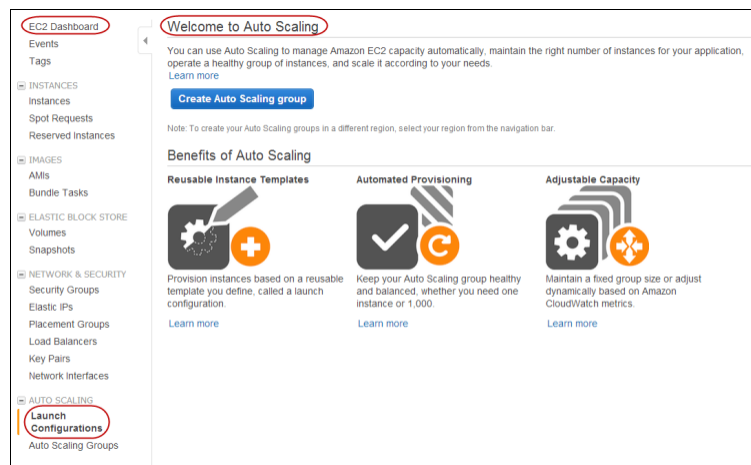
To create a launch configuration

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 console **Resources** page, in the left navigation pane, under **Auto Scaling**, click **Launch Configurations** to start the Auto Scaling wizard.
3. The Auto Scaling resources you create is tied to a region you specify and are not replicated across regions. For more information, see [Availability Zones and Regions \(p. 6\)](#).

The AWS Management Console selects a region for you by default. The default region is displayed in the navigation bar. If necessary, change the region. From the navigation bar, select the region that meets your needs.

This tutorial creates Auto Scaling resources in the US West (Oregon) region.

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 **Create Launch Configuration** wizard, the **1. 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. Notice that this configuration is marked **"Free tier eligible."**

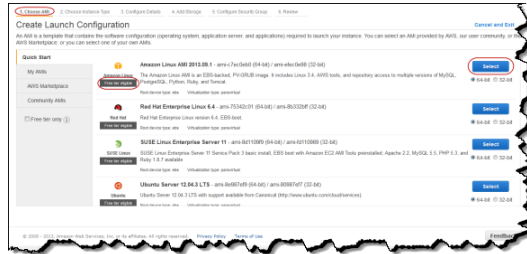
Note

When you sign up for AWS, you can test-drive some of the services and learn about AWS without charge. AWS calls this the [AWS Free Usage Tier](#). You are eligible to use the free usage tier for twelve months following your AWS sign-up date. The free tier includes 750 hours per month of Amazon EC2 Micro Instance usage.

If you are eligible for the Free Usage Tier and have not already exceeded the Free Usage Tier benefits for Amazon EC2, selecting an AMI marked **"Free tier eligible."** will not cost you anything to complete this tutorial. If you are not eligible for Free Usage tier or have exceeded Free Usage Tier benefits for Amazon EC2, you will be charged for the usage of the EC2 instance after the instance is launched. For a complete list of charges and specific prices for EC2 instances, see [Amazon EC2 Pricing](#).

Auto Scaling Developer Guide

Step 1: Create a Launch Configuration



- On the **2. Choose Instance Type** page, you can select the hardware configuration of your instance. This tutorial uses the `t1.micro` instance that is selected by default. Click **Next: Configure details** to let the **Create Launch Configuration** wizard complete other configuration settings for you, so you can get started quickly.
- On the **3. Configure Details** page, in the **Name** field, enter a name of your launch configuration (*my-first-1c*). Leave the other fields blank for this tutorial.
- In the next two steps you can add storage devices and configure a security group.

This tutorial uses the storage device that comes with the AMI and assumes that you have already created a security group before you started this tutorial. Click **Skip to Review**.

If you want to add additional storage devices and configure a security group, click **Next: Add Storage** and follow the instructions on the **4. Add Storage** and **5. Configure Security Group** pages.

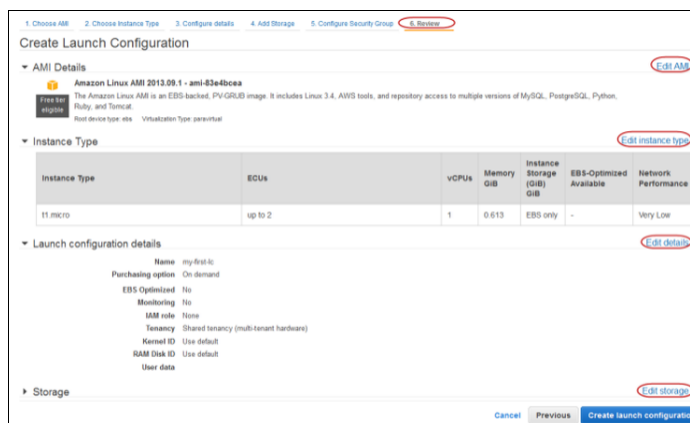
- On the **6. Review** page, review the details of your launch configuration.

Note

If you have not already selected a security group, the Launch Configuration wizard automatically defines the `AutoScaling-Security-Group-x` security group to allow you to connect to your instance. The `AutoScaling-Security-Group-x` security group enables all IP addresses (0.0.0.0/0) to access your instance over the specified ports.

If you would like to use a different security group, click **Edit security groups** on the bottom right, follow the instructions on the **5. Configure Security Group** page to either choose an existing security group or create a new one. Click **Review** to continue reviewing your launch configuration.

Similarly, if you want to change any other details, click **Edit details** to the right of the field you want to change.



- After you are done reviewing your launch configuration, click **Create launch configuration**.
- In the **Select an existing key pair or create a new key pair** dialog box, select one of the listed options. Or, select **Proceed without a key pair**. This option is acceptable for this short walkthrough.
- Select the acknowledgement check box.

Select an existing key pair or create a new key pair X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Proceed without a key pair

☒ acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI.

Cancel Create Launch Configuration

14. Click **Create Launch Configuration** to create your launch configuration.
15. 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**.

Step 2: Create an Auto Scaling Group

After you have created your launch configuration, you are ready to 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 **1. Configure Auto Scaling group details** page, enter the following details:
 - a. In the **Group name** field, enter a name for your Auto Scaling group *my-first-asg*.
 - b. Leave the **Group size** field set to the default value of 1 instance for this tutorial.
 - c. Leave the **Network** field blank for this tutorial.
 - d. Click the **Availability Zone(s)** field, and select *us-west-2a*.

Create Auto Scaling Group Cancel and Exit

Launch Configuration: my-first-lc

Group name: my-first-asg

Group size: Start with 1 instances

Network: Do not launch into a VPC Create new VPC

Availability Zone(s): us-west-2a

Advanced Details

Cancel Next: Configure scaling policies

2. Click **Next: Configure scaling policies**.
3. In the **Configure scaling policies** page, select **Keep this group at its initial size** for this tutorial.

If you want to configure scaling policies for your Auto Scaling group, see [Scaling Based on Metrics \(p. 60\)](#) and follow the instructions for creating scaling policies and CloudWatch alarms using the console.

Auto Scaling Developer Guide

Step 2: Create an Auto Scaling Group

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

Create Auto Scaling Group

You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to an Amazon CloudWatch alarm that you assign to it. In each policy, you can choose to add or remove a specific number of instances or a percentage of the existing group size, or you can set the group to an exact size. When the alarm triggers, it will execute the policy and adjust the size of your group accordingly. [Learn more](#) about scaling policies.

☒ Keep this group at its initial size

☐ Use scaling policies to adjust the capacity of this group

[Cancel](#) [Previous](#) [Next: Configure Notifications](#)

4. Click **Next: Configure Notifications**.

You can configure Auto Scaling to send notifications whenever your Auto Scaling group changes. You can either configure and add notifications now or configure your notifications later.

For this tutorial, skip this step and click **Next: Configure Tags**.

5. Tags help you organize your Auto Scaling groups. For more information, see [Add, Modify, or Remove Auto Scaling Group Tags](#) (p. 158).

Skip this step if you do not want to configure tags at this time.

To configure tags for your Auto Scaling group, complete the following steps:

- On the **4. Configure Tags** page, in the **Key** and **Value** boxes, enter the key (*environment*) and the value (*test*) for the tag.
- Keep the **Tag New Instances** box selected to propagate the tags to the instances launched by your Auto Scaling group.

Click the **Tag New Instances** box to avoid propagating the tags to the instances launched by your Auto Scaling group.

- Click **Add Tag** to add additional tags and then add additional keys and values.

6. When you have completed adding tags, click **Review**.

Review the details of your Auto Scaling group. You can click **Edit details** on the right to edit the details.

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

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

[Edit details](#)

Auto Scaling Group Details

Group name: my-first-arg
Group size: 1
Minimum Group Size: 1
Maximum Group Size: 1
Availability Zone(s): us-west-2a
Health Check Grace Period: 300
Detailed Monitoring: No

Scaling Policies [Edit scaling policies](#)

Notifications [Edit notifications](#)

Tags [Edit tags](#)

environment test tag new instances

[Cancel](#) [Previous](#) [Create Auto Scaling group](#)

7. Click **Create Auto Scaling Group**.

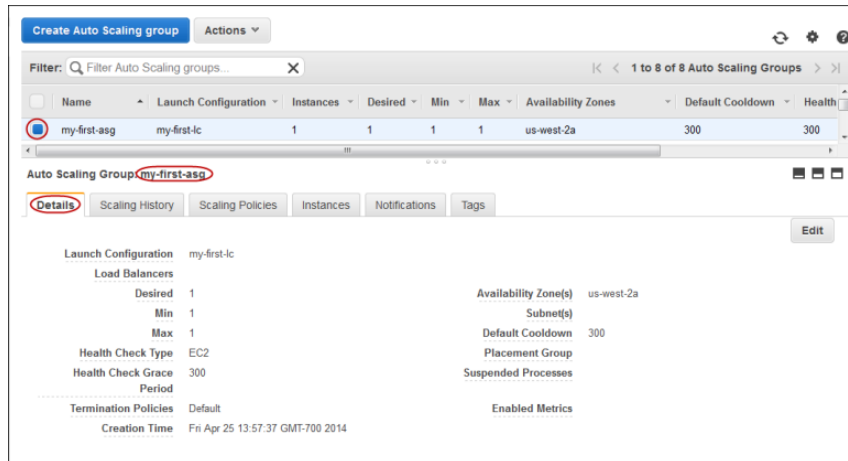
8. The **Auto Scaling Group creation status** page lets you know that your Auto Scaling group has been successfully created. Click **Close**.

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.

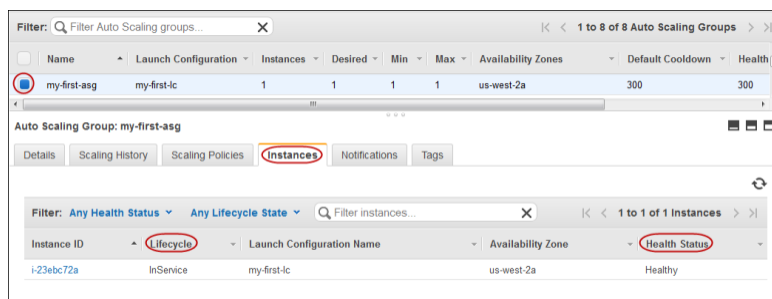
To verify that your Auto Scaling group has launched your EC2 instance

1. On the **Auto Scaling Groups** page, select the Auto Scaling group you just created.
2. The bottom pane displays the details of your Auto Scaling group. Make sure the **Details** tab is selected. If it is not, click **Details** tab.



3. Click the **Scaling History** tab. The **Status** column lets you know the current status of your instance. When your instance is launching, the status column shows `Not yet in service`. 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. In the bottom pane, click the **Instances** tab.
5. On the **Instances** view pane, you can view the current **Lifecycle** 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.



If you are eligible for free usage tier and are using the free usage tier AMI, you can skip the next step and continue running your `t1.micro` EC2 instance.

You can use this Auto Scaling set up as your base and do one of the following:

- [Maintaining a Fixed Number of Running EC2 Instances](#) (p. 53)

- [Manual Scaling \(p. 54\)](#)
- [Dynamic Scaling \(p. 57\)](#)
- [Getting Notifications When Your Auto Scaling Group Changes \(p. 248\)](#)

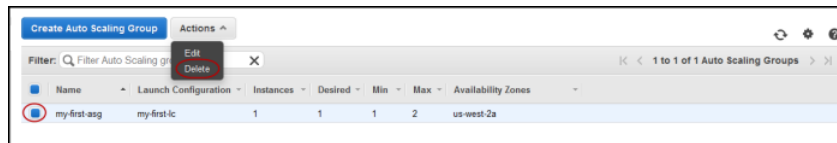
Go to the next step if you would like to delete your Auto Scaling set up.

Step 4: [Optional] Delete Your Auto Scaling Set Up

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. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group *my-first-asg*.
4. Click **Actions** and select **Delete**.



5. A confirmation dialog box opens. Click **Yes, Delete**.
6. The **Name** column indicates that the Auto Scaling group is being deleted. The **Desired**, **Min**, and **Max** columns show 0 instances for the *my-first-asg* Auto Scaling group.

Click the refresh button to see the message `No Auto Scaling groups found`. You have successfully deleted your Auto Scaling group. Skip the next step if you would like keep your launch configuration.

To delete your launch configuration

1. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Launch Configurations**.
2. On the **Launch Configurations** page, select your launch configuration *my-first-lc*.
3. Click **Actions** and select **Delete Launch Configuration**.
4. A confirmation dialog box opens. Click **Yes, Delete**.

Getting Started with Auto Scaling Using the CLI

This section walks you through the process of creating your basic Auto Scaling infrastructure within the EC2-Classic or default VPC platforms using the Auto Scaling command line interface (CLI) or Query API. By the end of this section, you will have:

- A launch configuration that Auto Scaling uses as template for the EC2 instances you want to launch. The template includes information about key pairs, security groups, and block device mapping, among other configuration settings.

- An Auto Scaling group that references the launch configuration.
- Verification that the Auto Scaling group is functioning.

If you are planning on using the CLI, be sure you have installed the tools. For information about installing the command line interface, see [Install the Auto Scaling CLI \(p. 16\)](#). For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs \(p. 23\)](#).

Topics

- [Using the CLI \(p. 35\)](#)
- [Using the Query API \(p. 39\)](#)

Using the CLI

Use the following Auto Scaling commands to create a launch configuration and Auto Scaling group, and to verify that your Auto Scaling group has been created.

Command	Description
<code>as-create-launch-config</code>	Creates a new launch configuration with the specified attributes.
<code>as-create-auto-scaling-group</code>	Creates a new Auto Scaling group with the specified name and other attributes.
<code>as-describe-auto-scaling-groups</code>	Describes the specified Auto Scaling groups if the group exists.
<code>as-describe-auto-scaling-instances</code>	Describes the specified instances. If the instances are not specified, Auto Scaling returns the description of all the instances associated with the AWS account.

Create a Launch Configuration

The launch configuration specifies the template that Auto Scaling uses to launch EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. In the following example, you use the `as-create-launch-config` CLI command. For more information, see [Launch Configuration](#).

The `as-create-launch-config` command takes the following arguments:

```
as-create-launch-config LaunchConfigurationName --image-id value --instance-  
type value [--associate-public-ip-address value][--spot-price value] [--iam-  
instance-profile value] [--block-device-mapping "key1=value1,key2=value2..." ]  
[--ebs-optimized] [--monitoring-enabled|--monitoring-disabled] [--kernel value  
] [--key value ] [--ramdisk value] [--group value[,value...] ] [--user-data  
value] [--user-data-file value] [General Options]
```

For a list of General Options commands you can use with any Auto Scaling command, see page 3 in the [Auto Scaling Quick Reference Card](#).

The only required options are the launch configuration name, image ID, and instance type. For this launch configuration, specify:

- Launch configuration name: `my-test-lc`
- Instance type: `m1.small`
- Image ID: `ami-0078da69`

Note

The AMI ID is provided for illustration purposes only. AMI IDs change over time. You can obtain current, valid AMI IDs by calling the `ec2-describe-images` CLI command.

If you purchased an Amazon EC2 Reserved Instance, and you want to apply the benefits to the instance launched by Auto Scaling, specify the instance criteria of your Reserved Instance in the launch configuration. For more information, see [Reserved Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Open a command prompt and enter the `as-create-launch-config` command.

```
as-create-launch-config my-test-lc --image-id ami-0078da69 --instance-type m1.small
```

If your request was successful, you should get a confirmation as in the following example:

```
OK-Created launch config
```

You now have a launch configuration called `my-test-lc` that launches an `m1.small` instance using the `ami-0078da69` AMI.

Create an Auto Scaling Group

After you have defined your launch configuration, you are ready to 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 can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions. For more information, see [Auto Scaling Groups](#).

To create an Auto Scaling group, use the `as-create-auto-scaling-group` CLI command. Alternatively, you can use the `CreateAutoScalingGroup` API call. For more information about the API call, see [CreateAutoScalingGroup](#) in the *Auto Scaling API Reference*.

The `as-create-auto-scaling-group` command takes the following arguments:

```
as-create-auto-scaling-group AutoScalingGroupName --availability-zones value[,value...] --launch-configuration value --max-size value --min-size value [--default-cooldown value] [--desired-capacity value] [--grace-period value] [--health-check-type value] [--load-balancers value[, value]] [--placement-group value] [--vpc-zone-identifier value] [General Options]
```

This command requires that you specify a name for your Auto Scaling group, a launch configuration, one or more Availability Zones, a minimum group size, and a maximum group size. The Availability Zones you choose determine the physical location of your Auto Scaling instances. The minimum and maximum group size tells Auto Scaling the minimum and maximum number of instances the Auto Scaling group should have.

Desired capacity is an important component of the `as-create-auto-scaling-group` command. Although it is an optional parameter, desired capacity tells Auto Scaling the number of instances you want to run initially. To adjust the number of instances you want running in your Auto Scaling group, you change the value of `--desired-capacity`. If you don't specify `--desired-capacity`, its value is the same as minimum group size.

If your AWS account supports the default virtual private cloud (default VPC) platform, it automatically comes with a default subnet in each Availability Zone, an Internet gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information about default VPC and Subnets, see [Your Default VPC and Subnets](#) in the *Amazon VPC User Guide*.

When your Auto Scaling group is launched within default VPC, it is automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the Auto Scaling group into the corresponding subnet for that Availability Zone. Alternatively, you can select the Availability Zone for your Auto Scaling group by selecting its corresponding default subnet.

For this launch configuration, specify the following options:

- Auto Scaling group name: `my-test-asg`
- Launch configuration name: `my-test-lc`
- [optional] Availability Zone: `us-east-1a`
- Minimum size: 1
- Maximum size: 10
- Desired capacity: 1

Note

If you purchased an Amazon EC2 Reserved Instance, and you want to apply the benefits to the instance launched by Auto Scaling, be sure to specify the Availability Zone of your Reserved Instance in your Auto Scaling group. For more information, see [Reserved Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you are launching an On-Demand instance, you will incur the standard Amazon EC2 usage fees for the instance until you terminate it as the last task in this tutorial. For more information about Amazon EC2 usage rates, see the [Amazon EC2 product page](#).

Enter the `as-create-auto-scaling-group` command as in the following example to launch your Auto Scaling group within EC2-Classic. If you are launching your Auto Scaling group within the default VPC, you need not specify the Availability Zone.

```
as-create-auto-scaling-group my-test-asg --launch-configuration my-test-lc --availability-zones us-east-1a --min-size 1 --max-size 10 --desired-capacity 1
```

If your request was successful, you should get a confirmation as in the following example:

```
OK-Created AutoScalingGroup
```

Based on the `my-test-asg` Auto Scaling group and the `my-test-lc` launch configuration, Auto Scaling will launch one EC2 instance in the `us-east-1a` Availability Zone.

Verify Your Auto Scaling Group Creation

You use the `as-describe-auto-scaling-groups` command to confirm that the `my-test-asg` Auto Scaling group exists. Use the `--headers` argument to print headings that describe each value that the command returns.

The `as-describe-auto-scaling-groups` command takes the following arguments:

```
as-describe-auto-scaling-groups [AutoScalingGroupNames [AutoScalingGroupNames...]] [--max-records value] [General Options]
```

Enter the `as-describe-auto-scaling-groups` command as in the following example:

```
as-describe-auto-scaling-groups my-test-asg --headers
```

If your request was successful, you should get the details of your group as in the following example:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1a	1	10	1

Verify Auto Scaling Instances

You can also use the `as-describe-auto-scaling-instances` command to check that the `my-test-asg` Auto Scaling group contains running instances. Use the `--headers` argument to print headings that describe each value that the command returns.

The `as-describe-auto-scaling-instances` command takes the following arguments:

```
as-describe-auto-scaling-instances [InstanceIds [InstanceIds...]] [--max-records value] [General Options]
```

Enter the `as-describe-auto-scaling-instances` command as in the following example:

```
as-describe-auto-scaling-instances --headers
```

If your request was successful, you should get the details of the launched instance like in the following example:

INSTANCE	INSTANCE-ID	GROUP-NAME	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG
INSTANCE	i-bcdd63d1	my-test-asg	us-east-1a	InService	HEALTHY	my-test-lc

Note

It may take a few minutes for the service to return the information.

Tasks Completed

You just performed the following tasks:

- Created a launch configuration
- Created an Auto Scaling group
- Confirmed that your Auto Scaling group exists
- Checked that your Auto Scaling group contains running instances

Following is the complete snippet used to perform these tasks. You can copy the snippet, replace the values with your own, and use the code to get started.

Note

The instance associated with the Auto Scaling group you just created is not launched instantly. If you run the snippet as a single code block, you won't see the instance information right away.

```
as-create-launch-config my-test-lc --image-id ami-0078da69 --instance-type m1.small
```

```
as-create-auto-scaling-group my-test-asg --launch-configuration my-test-lc --  
availability-zones us-east-1a --min-size 1 --max-size 10 --desired-capacity 1  
as-describe-auto-scaling-groups --headers  
as-describe-auto-scaling-instances --headers
```

Using the Query API

Use the following Auto Scaling actions to create a launch configuration and Auto Scaling group, and to verify that your Auto Scaling group has been created.

Command	Description
CreateLaunchConfiguration	Creates a new launch configuration with specified attributes.
CreateAutoScalingGroup	Creates a new Auto Scaling group with a specified name and other attributes.
DescribeAutoScalingGroups	Describes the specified Auto Scaling groups if the group exists.
DescribeAutoScalingInstances	Describes the specified instances. If the instances are not specified, Auto Scaling returns the description of all the instances associated with the AWS account.

For more information about Auto Scaling actions, see [Auto Scaling API Reference](#).

Create a Launch Configuration

The launch configuration specifies the template that Auto Scaling uses to launch EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. For more information, see [Launch Configuration](#).

Call the `CreateLaunchConfiguration` action by specifying the following parameters:

- Launch configuration name: `my-test-lc`
- Instance type: `m1.small`
- Image ID: `ami-0078da69`

Note

The AMI ID is provided for illustration purposes only. AMI IDs change over time. You can obtain current, valid AMI IDs by calling the Amazon EC2 [DescribeImages](#) action.

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?LaunchConfigurationName=my-test-lc  
&ImageId=ami-0078da69  
&InstanceType=m1.small  
&Action=CreateLaunchConfiguration  
&AUTHPARAMS
```

If your request was successful, you should get a confirmation like in the following example:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7c6e177f-f082-11e1-ac58-3714bEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

You now have a launch configuration called `my-test-lc` that launches an `m1.small` instance using the `ami-0078da69` AMI.

Create an Auto Scaling Group

After you have defined your launch configuration, you are ready to 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 can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions. For more information, see [Auto Scaling Groups](#).

To create an Auto Scaling group, you must specify a name for your group, a launch configuration, one or more Availability Zones, a minimum group size, and a maximum group size. The Availability Zones you choose determine the physical location of your Auto Scaling instances. The minimum and maximum group size tells Auto Scaling the minimum and maximum number of instances the Auto Scaling group should have.

Desired capacity is an important component of the Auto Scaling group creation process. Although it is an optional parameter, desired capacity tells Auto Scaling the number of instances you want to run initially. To adjust the number of instances you want running in your Auto Scaling group, you change the value of `DesiredCapacity`. If you don't specify the desired capacity for your Auto Scaling group, its value is the same as minimum group size.

If your AWS account supports the default virtual private cloud (default VPC) platform, it automatically comes with a default subnet in each Availability Zone, an Internet gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information about default VPCs and subnets, see [Your Default VPC and Subnets](#) in the *Amazon VPC User Guide*.

When your Auto Scaling group is launched within a default VPC, it is automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the Auto Scaling group into the corresponding subnet for that Availability Zone. Alternatively, you can select the Availability Zone for your Auto Scaling group by selecting its corresponding default subnet.

Call the `CreateAutoScalingGroup` action by specifying the following parameters:

- Auto Scaling group name: `my-test-asg`
- Launch configuration name: `my-test-lc`
- [optional] Availability Zone: `us-east-1a`
- Minimum size: 1
- Maximum size: 10
- Desired capacity: 1

Important

You will incur the standard Amazon EC2 usage fees for the instance until you terminate it as the last task in this tutorial. For more information about Amazon EC2 usage rates, see the [Amazon EC2 product page](#).

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&AvailabilityZones.member.1=us-east-1a
&MinSize=1
&MaxSize=10
&DesiredCapacity=1
&LaunchConfigurationName=my-test-lc
&Action=CreateAutoScalingGroup
&AUTHPARAMS
```

If your request is successful, you should get a confirmation like the following example:

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>8d798a29-f083-11e1-bdfb-cb223EXAMPLE</RequestId>
  </ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

Based on the `my-test-asg` Auto Scaling group and the `my-test-lc` launch configuration, Auto Scaling will launch one EC2 instance in the `us-east-1a` Availability Zone.

Verify Auto Scaling Group Creation

You can confirm if Auto Scaling has launched an EC2 instance using the `my-test-lc` launch configuration in Availability Zone `us-east-1a` by looking at the description of your Auto Scaling group `my-test-asg`.

Call the `DescribeAutoScalingGroups` action by specifying the following parameter:

- Auto Scaling group name: `my-test-asg`

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-asg
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The response includes details about the group and instance launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-02-12T22:14:49.235Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
</DescribeAutoScalingGroupsResponse>
```

```
<Instances>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1a</AvailabilityZone>
    <InstanceId>i-6fecd61f</InstanceId>
    <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
    <LifecycleState>InService</LifecycleState>
  </member>
</Instances>
<DesiredCapacity>1</DesiredCapacity>
<AvailabilityZones>
  <member>us-east-1a</member>
</AvailabilityZones>
<LoadBalancerNames/>
<MinSize>1</MinSize>
<VPCZoneIdentifier/>
<HealthCheckGracePeriod>0</HealthCheckGracePeriod>
<DefaultCooldown>300</DefaultCooldown>
<AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:auto
ScalingGroup:5d1ee7f3-f0f6-42bd-851e-0513f88c56b0:autoScalingGroupName/my-test-
asg</AutoScalingGroupARN>
<TerminationPolicies>
  <member>Default</member>
</TerminationPolicies>
<MaxSize>10</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>e57b79d1-7564-11e2-9320-f7b1aEXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Verify Auto Scaling Instances

You can also confirm if Auto Scaling has launched your instance by seeing the description of all running instances associated with your AWS account.

Call the `DescribeAutoScalingInstances` action.

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?MaxRecords=20
&Action=DescribeAutoScalingInstances
&AUTHPARAMS
```

The response includes details about the instance launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
```

```
<AvailabilityZone>us-east-1a</AvailabilityZone>
<InstanceId>i-6fec61f</InstanceId>
<LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
<LifecycleState>InService</LifecycleState>
</member>
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>f6e01d93-7567-11e2-90b3-8dedfEXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

Note

It may take a few minutes for the service to return the information.

Tasks Completed

You just performed the following tasks:

- Created a launch configuration
- Created an Auto Scaling group
- Confirmed that your Auto Scaling group exists
- Checked that your Auto Scaling group contains running instances

Planning your Auto Scaling Group

Auto Scaling, when correctly implemented, can provide 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. 45). 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. 46) and [termination policies](#) (p. 49), 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. 105), which describes the role [AWS Identity and Access Management](#) plays in managing your EC2 instances in an Auto Scaling group.

Topics

- [Scaling the Size of Your Auto Scaling Group](#) (p. 45)

- [Controlling Access to Your Auto Scaling Resources](#) (p. 105)
- [Creating Launch Configurations](#) (p. 111)
- [Creating Your Auto Scaling Groups](#) (p. 122)
- [Controlling How Instances Launch and Terminate](#) (p. 137)
- [Add, Modify, or Remove Auto Scaling Group Tags](#) (p. 158)
- [Launching Spot Instances in Your Auto Scaling Group](#) (p. 162)

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. Some considerations you should keep 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.

For more information about other issues to review about your application, see [Planning your Auto Scaling Group](#) (p. 44).

The following topics describe some of the ways in which you can add Auto Scaling to your architecture, including information on two very important Auto Scaling features: cooldowns and termination policies. [Cooldowns](#) (p. 46) are periods of time during which Auto Scaling ignores any additional scaling actions. [Termination policies](#) (p. 49) are criteria that determine which instances Auto Scaling should terminate first. Understanding these two features can help you make sure your Auto Scaling group performs as expected.

Learn more about different Auto Scaling implementations:

- [Maintaining a Fixed Number of Running EC2 Instances](#) (p. 53)

Use this scaling plan if you would like Auto Scaling to maintain the minimum or specified number of instances in your Auto Scaling group at all times.

- [Manual Scaling](#) (p. 54)

Use this scaling plan if you would like to change the number of running instances in your Auto Scaling group manually at any time.

- [Dynamic Scaling](#) (p. 57)

Use this scaling plan if you need to scale dynamically in response to changes in the demand for your application. When you scale based on demand, you must specify when and how to scale.

- [Scheduled Scaling](#) (p. 78)

Use this scaling plan if you want to scale your application on a predefined schedule. You can specify the schedule for scaling one time only or provide details for scaling 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, say 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 scale out criteria for both policies are met at the same time, Auto Scaling will give 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 prompts 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 \(p. 54\)](#) or [dynamic scaling \(p. 57\)](#). 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.

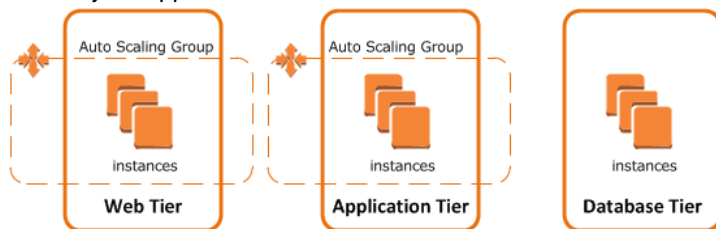
The Auto Scaling cooldown period is a configurable setting that determines when Auto Scaling should suspend any scaling activities related to a specific Auto Scaling group. This cooldown period is important, because it helps to ensure you don't launch or terminate more resources than you need.

Topics

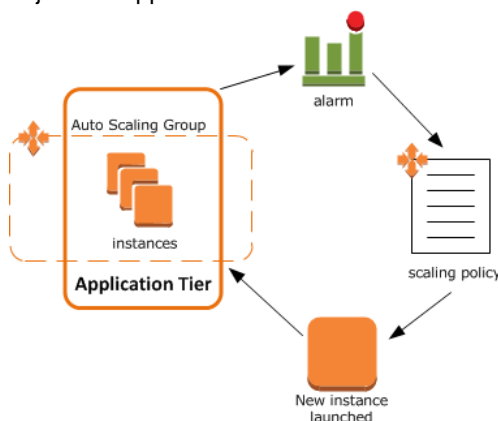
- [Example: Auto Scaling Cooldowns \(p. 47\)](#)
- [Default Cooldowns \(p. 47\)](#)
- [Scaling-Specific Cooldowns \(p. 48\)](#)
- [Cooldowns and Multiple Instances \(p. 49\)](#)
- [Cooldowns and Lifecycle Hooks \(p. 49\)](#)
- [Cooldowns and Spot Instances \(p. 49\)](#)

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.



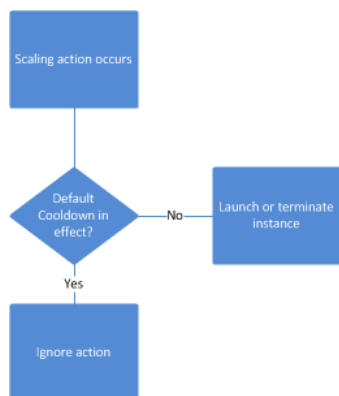
These instances use a configuration script to install and configure software before the instance is put into service. As a result, it takes around two or three minutes from the time the instance launches to when it is in service. (The actual time, of course, depends on several factors, such as whether 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. 48) 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 \(p. 57\)](#) actions, as described the preceding section, but [manual scaling \(p. 54\)](#) 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 ([aws autoscaling create-auto-scaling-group](#))
- Auto Scaling CLI (`as-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 ([aws autoscaling update-auto-scaling-group](#))
- Auto Scaling CLI (`as-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 \(p. 47\)](#) 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” \(p. 47\)](#). 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 ([aws autoscaling put-scaling-policy](#))
- Auto Scaling CLI (`as-auto-scaling-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. 137) 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. 162) 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

With each Auto Scaling group, you control when to add 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, and you can automate the process through the use of a scaling policy.

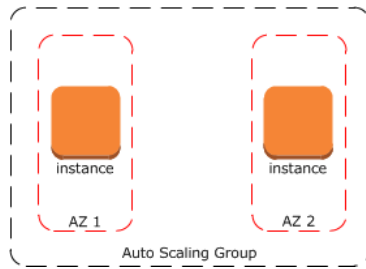
When you have Auto Scaling automatically scale in, you need to decide which instances should terminate first. You can configure this through the use of a termination policy.

Topics

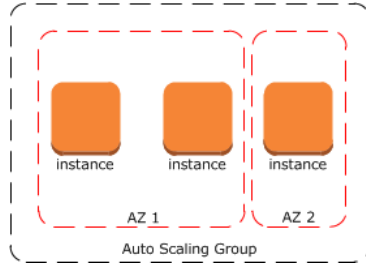
- [Default Termination Policy](#) (p. 49)
- [Customizing the Termination Policy](#) (p. 51)

Default Termination Policy

Consider an Auto Scaling group that has a desired capacity of two instances, and scaling policies that increase and decrease the number of instances by 1 when certain thresholds are met.

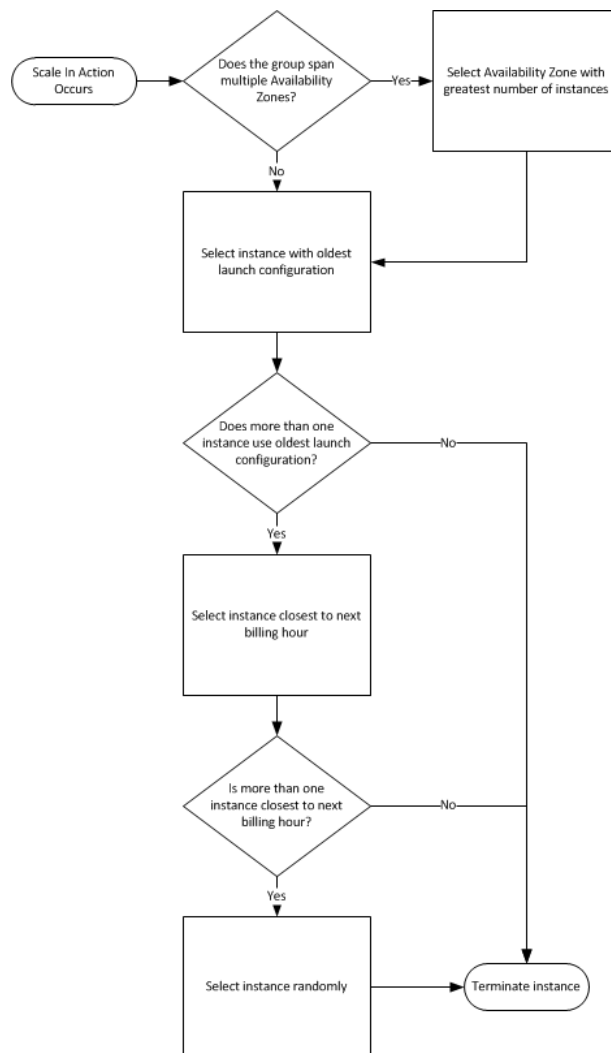


An increase in traffic occurs, which results in the Auto Scaling group launching a new instance. The Auto Scaling group now has three instances in it.



After a period of time, the amount of traffic subsides, and the scale in policy takes effect. This results in Auto Scaling terminating one of the instances. Because this group does not have a specific termination policy assigned to it, Auto Scaling uses the default termination policy. This means that:

1. Auto Scaling first checks to see if it has instances in multiple Availability Zones. If one Availability Zone has more instances in it than another, it terminates an instance from that Availability Zone. If both Availability Zones have an equal number of instances, Auto Scaling selects a random Availability Zone from which to terminate an instance.
2. Auto Scaling next looks to see which instance in the Availability Zone uses the oldest launch configuration. Auto Scaling makes this determination by using `DescribeLaunchConfigurations` and using the value in the `CreatedTime` field.
3. If Auto Scaling detects that more than one instance use the oldest launch configuration, it identifies which instance is closest to the next billing hour, and terminates that instance.
4. If more than one instance is closest to the next billing hour, Auto Scaling selects the instance randomly.



This default termination policy helps ensure that your network architecture spans Availability Zones evenly. By selecting an instance closest to the next billing hour, Auto Scaling also helps you maximize the use of your EC2 instances while minimizing the number of hours you are billed for Amazon EC2 usage.

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 selects an Availability Zone at random and applies the termination policy that you specified.

Auto Scaling currently supports the following custom termination policies:

- **OldestInstance.** With this policy in place, 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.** This policy instructs Auto Scaling to terminate the newest instance in the Auto Scaling group. You might use this policy if you are testing a new launch configuration but don't want to keep the new configuration in production.
- **OldestLaunchConfiguration.** Use this termination policy when you want to Auto Scaling to terminate instances that have the oldest launch configuration. This policy can help when you're updating an Auto Scaling group and want to phase out instances using a previous configuration.
- **ClosestToNextInstanceHour.** This policy helps you run your Auto Scaling group in a cost-effective manner. With this policy in place, Auto Scaling terminates instances that are closest to the next billing hour. This allows you to maximize your use of the EC2 instances in your Auto Scaling group.
- **Default.** This policy instructs Auto Scaling to use its default termination policy. You might use this policy when you have more than one scaling policy associated with a given 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, the following AWS CLI command updates 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"
```

Note

If you use the default termination policy, make sure it's the last one in the list of termination policies. For example:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-  
asg --termination-policies "OldestLaunchConfiguration,Default"
```

You can customize a termination policy for an Auto Scaling group using the AWS Management Console, AWS CLI, or the Auto Scaling API.

To customize a termination policy with the console:

1. Create the Auto Scaling group. For more information, see [Creating Your Auto Scaling Groups \(p. 122\)](#).
2. From the EC2 Dashboard, click **Auto Scaling Groups**.
3. Select the group to update.
4. From the **Actions** menu, select **Edit**.
5. On the **Details** tab, locate the **Termination Policies** field. Select one or more termination policies.
6. Click **Save**.

To customize a termination policy using the AWS CLI, use one of the following commands:

- `aws autoscaling create-auto-scaling-group`
- `aws autoscaling update-auto-scaling-group`

To customize a termination policy using the Auto Scaling API, use one of the following actions:

- `CreateAutoScalingGroup`
- `UpdateAutoScalingGroup`

Maintaining a Fixed Number of Running EC2 Instances

Topics

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

To maintain the same instance level, 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.

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 description shows any other state other than `running` or if the system status description shows `impaired`, Auto Scaling considers the instance to be unhealthy and launches a replacement.

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 calls to the Amazon EC2 action [DescribeInstanceStatus](#) returns any other state other than `running`, the system status shows `impaired`, or the calls to Elastic Load Balancing action [DescribeInstanceHealth](#) returns `OutOfService` in the instance state field.

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.

To learn more about EC2 instance status checks, see [Monitoring the Status of Your Instances](#) in the *Amazon EC2 User Guide for Linux Instances*. To learn more about Elastic Load Balancing health checks, see [Elastic Load Balancing Health Check](#) in the *Elastic Load Balancing Developer Guide*.

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 Processes](#).

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.

Follow the instructions in the [Getting Started with Auto Scaling \(p. 28\)](#) tutorial to step through the following activities:

1. Create a launch configuration (for example, `my-test-1c`).
2. Use the launch configuration to create an Auto Scaling group by specifying the number of running instances you want the Auto Scaling group to have at any point of time.

3. Verify the creation of your Auto Scaling group and the launching of the instances in your Auto Scaling group.
4. Verify the health state of the instances in the Auto Scaling group

Manual Scaling

At any time, you can manually change the size of an existing Auto Scaling group. You only need to specify a change to 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 group size. You can change the size of your Auto Scaling group using AWS Management Console, the Auto Scaling command line interface (CLI), or the Query API.

Topics

- [Scaling Manually Using the Console \(p. 54\)](#)
- [Scaling Manually Using the Command Line Interface \(p. 54\)](#)
- [Scaling Manually Using the Query API \(p. 55\)](#)

Scaling Manually Using the Console

To change the size of your Auto Scaling group manually

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the **Auto Scaling Groups** page, select your Auto Scaling group (*my-test-asg*) from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab and click **Edit**.
5. In the **Desired** field, enter *2* and click **Save**.

In the next step, verify that your Auto Scaling group has launched *1* 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. Click the refresh button to 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 Command Line Interface

Use the `as-set-desired-capacity` command to change the size of your Auto Scaling group, as shown in the following example:

```
as-set-desired-capacity my-test-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 \(p. 46\)](#).

Auto Scaling returns the following response:

```
OK-Desired Capacity Set
```

Use the `as-describe-auto-scaling-groups` command to confirm that the size of your Auto Scaling group has changed, as in the following example:

```
as-describe-auto-scaling-groups my-test-asg --headers
```

Note

Specify the `--headers` general option to show column headers that organize the `as-describe-auto-scaling-groups` command information.

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

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1e	1	2	5	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-98e204e8	us-east-1e	InService	Healthy	my-test-lc		
INSTANCE	i-2a77ae5a	us-east-1e	InService	Healthy	my-test-lc		

The desired capacity of your Auto Scaling group shows the new value. Your Auto Scaling group has launched an additional instance.

Scaling Manually Using the Query API

Use the [SetDesiredCapacity](#) action with the following parameters to change the size of your Auto Scaling group:

- `AutoScalingGroupName = my-test-asg`
- `DesiredCapacity = 2`
- `HonorCooldown = True`

By default, the command overrides any cooldown period specified for the Auto Scaling group. Set `HonorCooldown` to `True` if you want Auto Scaling to reject the default behavior and honor the cooldown period. For more information, see [Understanding Auto Scaling Cooldowns \(p. 46\)](#).

If your request is successful, you should get a confirmation as in the following example:

```
<SetDesiredCapacityResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>9fb7e2db-6998-11e2-a985-57c82EXAMPLE</RequestId>
  </ResponseMetadata>
</SetDesiredCapacityResponse>
```

Use the [DescribeAutoScalingGroups](#) action with the following parameter to confirm that the size of your Auto Scaling group has changed.

- AutoScalingGroupName = my-test-asg

If your request is successful, you should get a confirmation as in the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-28T22:14:05.886Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-42b66e32</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-88ae76f8</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:auto
ScalingGroup:fec2667a-0410-419e-a6fa-16f37Example:
autoScalingGroupName/my-test-asg2</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
</DescribeAutoScalingGroupsResponse>
```

```
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
    </member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
    <RequestId>e79c8299-699a-11e2-b287-b79f0EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

The desired capacity of your Auto Scaling group shows the new value. Your Auto Scaling group has launched an additional instance.

Dynamic Scaling

Topics

- [Architectural Overview of Dynamic Scaling \(p. 58\)](#)
- [Scaling Based on Metrics \(p. 60\)](#)
- [Scheduled Scaling \(p. 78\)](#)
- [Scaling Based on Amazon SQS \(p. 88\)](#)

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 *CloudWatch Developer Guide*.

You use Auto Scaling to create your scaling policies. When a scaling policy is executed, it changes the current size of your Auto Scaling group by the amount you specify 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.

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 the policy name, the name of the Auto Scaling group to associate the policy with, and the following parameters:

- **ScalingAdjustment**— The number of instances by which to scale.
- **AdjustmentType**— Indicates whether the `ScalingAdjustment` is an absolute value, a constant increment, or a percentage of the current capacity.

A positive adjustment value increases the current capacity and a negative adjustment value decreases the current capacity.

Auto Scaling supports the following adjustment types:

- **ChangeInCapacity**: Use this to increase or decrease 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**: Use this to change the current capacity of your Auto Scaling group to the exact value specified. For example, the capacity of your Auto Scaling group is set to five instances, and you then create a scaling policy on your Auto Scaling group, specify the type as `ExactCapacity`, and the adjustment as three. 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**: Use this to increase or decrease the desired capacity by a percentage of the desired capacity. For example, let's say that the desired capacity of your Auto Scaling group is set to 10 instances. You then create a scaling policy on your Auto Scaling group, specify the type as `PercentChangeInCapacity`, and the adjustment as ten. When the policy is executed, your Auto Scaling group has eleven running instances because 10 percent of 10 instances is 1 instance, and 1 instance plus 10 instances equals 11 instances.

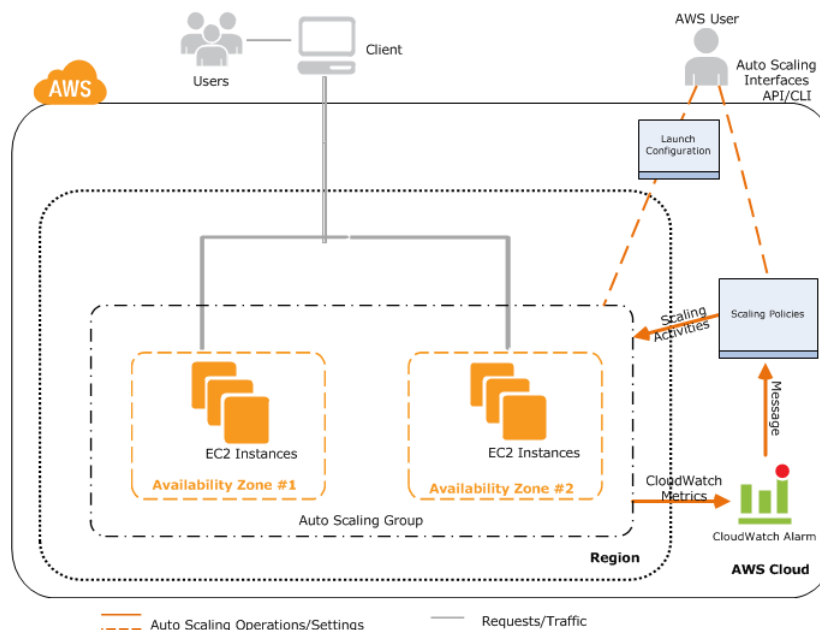
Auto Scaling rounds off the values returned by the `PercentChangeInCapacity` in one of the following ways:

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

Auto Scaling does not scale above the maximum size or below the minimum size of the Auto Scaling group.

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.



This section provides a quick walkthrough of the flow of events illustrated in the architectural diagram

, and assumes that the AWS user in the diagram has signed up to use AWS, is familiar with using EC2 instances, and can set up the CloudWatch metrics and alarms. This walkthrough also assumes that 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 a 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.

These events begin when a client sends a request to an 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

This section walks you through the process for creating Auto Scaling policies and CloudWatch alarms. You create CloudWatch alarms that watch over the scale-in and scale-out metrics that you specified when you created your policy. 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.

The following steps outline how to create policies and alarms.

1. Create a launch configuration
2. Create an Auto Scaling group
3. Create two policies, one for scaling out and one for scaling in.
4. Create CloudWatch alarms to watch over metrics that Auto Scaling sends to CloudWatch. This walkthrough uses the *CPUUtilization* metrics.
5. Verify that the policies and alarms have been created.

Topics

- [Scaling with Metrics Using the AWS Management Console \(p. 60\)](#)
- [Scaling with Metrics Using the Command Line Interface \(p. 64\)](#)
- [Scaling with Metrics Using the Query API \(p. 70\)](#)

Scaling with Metrics Using the AWS Management Console

In this section, you use the AWS Management Console to configure scaling policies for your Auto Scaling group.

Topics

- [Create an Auto Scaling Group With Scaling Policies and CloudWatch Alarms \(p. 60\)](#)
- [Create Scaling Policies \(p. 61\)](#)
- [Verify Your Scaling Policies and CloudWatch Alarms \(p. 63\)](#)

Create an Auto Scaling Group With Scaling Policies and CloudWatch Alarms

In this section, you create an Auto Scaling group *my-test-asg*, create two scaling policies that tell the Auto Scaling group how to scale and you create CloudWatch alarms to watch over the *CPUUtilization* metrics of the Auto Scaling group.

To create Auto Scaling group

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Launch Configurations**.
3. On the **Launch Configurations** page, select *my-test-lc*.

Note

If you do not already have a launch configuration or want to use a new launch configuration for this procedure, click **Create Auto Scaling Group** or **Create Launch Configuration** and follow the instructions mentioned in the [Create Launch Configuration \(p. 29\)](#) to create a new launch configuration.

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 *my-test-asg*.
 - b. In the **Group size** field, enter the number of instances you want your Auto Scaling group to start with *2*.
 - c. Leave the **Network** field blank for this procedure.
 - d. Click the **Availability Zone(s)** dialog box, and select an Availability Zone *us-west-2a*.

- e. Click **Next: Configure scaling policies**.

Create Scaling Policies

In this section, you'll create two scaling policies, *my-scaleout-policy*, which increases the capacity of the *my-test-asg* group by 30 percent of its size, and *my-scalein-policy*, which decreases the capacity of *my-test-asg* 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.

1. To create scaling policies

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** dialog box 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** dialog box, 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).

Note

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](#).

- 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. You can leave the defaults or define your own. For this example, monitor for 1 period of 5 minutes.

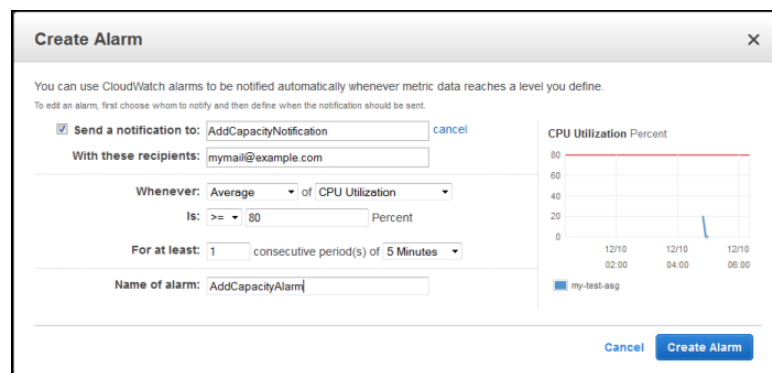
Note

A shorter period creates a more sensitive alarm. A longer period can mitigate brief spikes in a metric.

- In **Name of alarm**, a name is automatically generated for you. You can type in the field to change the name.

Important

You cannot modify the name after you create the alarm.



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-scalein-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. You can leave the defaults or define your own. For this example, monitor for 1 period of 5 minutes.

Note

A shorter period creates a more sensitive alarm. A longer period can mitigate brief spikes in a metric.

- In the **Name of alarm** field, a name is automatically generated for you. You can change the name by typing in a new name.

Important

You cannot modify the name after you create the alarm.

- In the **Take the action:** dialog boxes, select `Remove`, enter `2`, click and select `instances`.
- 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-arg

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-arg

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, review the details of your Auto Scaling group and scaling policies. You can click **Edit** on the right to edit the details.

1. Configure Auto Scaling Group Details 2. Configure Scaling Policies 3. Configure Notifications 4. Review

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling Group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details [Edit details](#)

Name: my-test-arg

Desired Capacity: 2

Minimum Group Size: 1

Maximum Group Size: 6

Availability Zone(s): us-west-2a

Health Check Grace Period: 300

Detailed Monitoring: No

Scaling Policies [Edit scaling policies](#)

my-scaleout-policy With alarm = AddCapacity: Add 30 Percent of group with minimum adjustment of 2 and 300 seconds between activities

my-scalein-policy With alarm = RemoveCapacity: Remove 2 instances and 300 seconds between activities

Notifications [Edit notifications](#)

[Cancel](#) [Previous](#) [Create Auto Scaling Group](#)

2. Click **Create Auto Scaling Group**.
3. The **Auto Scaling Group creation status** page lets you know that your Auto Scaling group was successfully created. Click **View your Auto Scaling Groups**.
4. On the **Auto Scaling Groups** page, select *my-test-asg*, the Auto Scaling group you just created.
5. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab.
6. Click the **Scaling History** tab. The **Status** column lets you know that your Auto Scaling group has successfully launched 2 instances.
7. 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.

8. Click the **Scaling Policies** tab

to can view the description of the policies created for the Auto Scaling group.

Scaling with Metrics Using the Command Line Interface

Before you begin, be sure you have installed the Auto Scaling CLI and the CloudWatch command line tools. For information about installing Auto Scaling CLI, see [Install the Auto Scaling CLI \(p. 16\)](#). For information about installing CloudWatch tools, see [Command Line Tools](#).

Use the following Auto Scaling commands to create a launch configuration, Auto Scaling group and Auto Scaling policies.

Command	Description
<code>as-create-launch-config</code>	Creates a new launch configuration with specified attributes.
<code>as-create-auto-scaling-group</code>	Creates a new Auto Scaling group with a specified name and other attributes.
<code>as-describe-auto-scaling-groups</code>	Describes the Auto Scaling groups, if the groups exist.
<code>as-put-scaling-policy</code>	Creates an Auto Scaling policy.
<code>as-describe-policies</code>	Describes the policies associated with your account.

For more information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the [Auto Scaling Quick Reference Card](#).

Use the following CloudWatch commands to create a CloudWatch alarm and retrieve alarm details from it.

Command	Description
<code>mon-put-metric-alarm</code>	Creates an alarm and associates it with the specified CloudWatch metric and the Auto Scaling policy.

Command	Description
<code>mon-describe-alarms</code>	Retrieves the details of the specified alarm. If no alarm is specified, retrieves details of all the alarms for the AWS account.

For more information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the [Amazon CloudWatch Command Line Reference](#).

Topics

- [Create Launch Configuration \(p. 65\)](#)
- [Create an Auto Scaling Group \(p. 65\)](#)
- [Verify Your Auto Scaling Group \(p. 66\)](#)
- [Create Scaling Policies \(p. 66\)](#)
- [Create CloudWatch Alarms \(p. 68\)](#)
- [Verify Your Scaling Policies and CloudWatch Alarms \(p. 69\)](#)

Create Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the [Basic Scenario in Auto Scaling](#).

For this procedure, specify the following values for the `as-create-launch-config` command:

- Launch configuration name = `my-test-lc`
- Image ID = `ami-514ac838`

If you don't have an AMI, and you want to find a suitable one, see [Finding a Suitable AMI](#) in the Amazon EC2 User Guide for Linux Instances.

- Instance type = `m1.small`

Your command should look similar to the following example:

```
as-create-launch-config my-test-lc --image-id ami-514ac838 --instance-type m1.small
```

If your request was successful, you should get a confirmation like the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Use the `as-create-auto-scaling-group` command by specifying the following values:

- Auto Scaling group name = `my-test-asg`
- Launch configuration name = `my-test-lc`
- Availability Zone = `us-east-1e`
- Max size = 5
- Min size = 1

Your command should look like the following example:

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

If your request was successful, you should get a confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Verify Your Auto Scaling Group

Use the `as-describe-auto-scaling-groups` command, as in the following example, to verify your Auto Scaling group.

```
as-describe-auto-scaling-groups my-test-asg --headers
```

Note

Specify the `--headers` general option to show column headers that organize the `as-describe-auto-scaling-groups` command's information.

Auto Scaling responds with details about the group and instances launched. The information you get should be similar to the following example.

```
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG  AVAILABILITY-ZONES  MIN-SIZE
MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  my-test-asg  my-test-lc      us-east-1e          1
5                1                Default
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE  STATUS  LAUNCH-CONFIG
INSTANCE  i-cbd7caba   us-east-1e        InService  InService  my-test-lc
```

You can see that Auto Scaling launched an instance using the `my-test-lc`. It is healthy, and running (InService).

Create Scaling Policies

Scaling policies tells the Auto Scaling group what to do when the specified conditions change.

In this procedure, you'll create two scaling policies, `my-scaleout-policy`, which increases the capacity of the `my-test-asg` group by 30 percent of its size, and `my-scalein-policy`, which decreases the capacity of `my-test-asg` group to two instances.

To create scaling policies

1. Use the `as-put-scaling-policy` command by specifying the following values:

- Policy name = `my-scaleout-policy`
- Auto Scaling group name = `my-test-asg`
- Adjustment = 30
- Adjustment type = `PercentChangeInCapacity`

Your command should look similar to the following example:

```
as-put-scaling-policy my-scaleout-policy --auto-scaling-group my-test-asg  
--adjustment=30 --type PercentChangeInCapacity
```

Note

No Auto Scaling name, including policy names, can contain the colon (:) character because colons serve as delimiters in Amazon Resource Names (ARNs).

2. Auto Scaling returns the ARN that serves as a unique name for the new policy. Subsequently, you can use either the ARN or a combination of the policy name and group name to specify the policy.

```
arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:ac542982-cbeb-4294-  
891c-a5a941dfa787:autoScalingGroupName/ my-test-asg:policyName/my-scaleout-  
policy
```

Copy the ARN in a safe place. You'll need it to create CloudWatch alarms.

3. Use the `as-put-scaling-policy` command to create another policy by specifying the following values:
 - Policy name = `my-scalein-policy`
 - Auto Scaling group name = `my-test-asg`
 - Adjustment = `-2`
 - Adjustment type = `ChangeInCapacity`

Your command should look similar to the following example:

```
as-put-scaling-policy my-scalein-policy --auto-scaling-group my-test-asg --  
adjustment=-2 --type ChangeInCapacity
```

Note

If you are running Windows, you must use quotation marks when specifying the adjustment value, for example `--adjustment=-2`.

4. Auto Scaling returns the ARN for the policy.

```
arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:4ee9e543-86b5-4121-  
b53b-aa4c23b5bbcc:autoScalingGroupName/ my-test-asg:policyName/my-scalein-  
policy
```

Copy the ARN in a safe place. You'll need it to create CloudWatch alarms.

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. To use a different cooldown period than the default specified in the Auto Scaling group, use the `--cooldown cooldown value` option with the `as-put-scaling-policy` command. 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 \(p. 46\)](#).

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.

Before you begin, be sure you have installed the CloudWatch CLI. For more information, see [Command Line Tools](#).

To create CloudWatch alarms

1. Use the CloudWatch command `mon-put-metric-alarm` to create an alarm to for increasing the size of the Auto Scaling group when the average CPU usage of all the instances goes up to 80 percent by specifying the following values:
 - Alarm name = `AddCapacity`
 - Metric name = `CPUUtilization`
 - Namespace = `"AWS/EC2"`
 - Statistic = `Average`
 - Period = `120`
 - Threshold = `80`
 - Comparison operator = `GreaterThanOrEqualToThreshold`
 - Dimensions = `"AutoScalingGroupName=my-test-asg"`
 - Evaluation periods = `2`
 - Alarm action = `arn:aws:autoscaling:us-east-1:123456789012:scaling-Policy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/ my-test-asg:policyName/my-scaleout-policy`

For more information about the options used, see [mon-put-metric-alarm](#) in the *Amazon CloudWatch Developer Guide*.

Your command should look like the following example:

```
prompt>mon-put-metric-alarm --alarm-name AddCapacity --metric-name
CPUUtilization --namespace AWS/EC2
--statistic Average --period 120 --threshold 80 --comparison-operator
GreaterThanOrEqualToThreshold --dimensions AutoScalingGroupName=my-test-asg

--evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScal
ingGroupName/ my-test-asg:policyName/my-scaleout-policy
```

If your request was successful, you should get a confirmation that looks like the following example:

```
OK-Created Alarm
```

2. Use the CloudWatch command `mon-put-metric-alarm` to create an alarm for decreasing the size of the Auto Scaling group when the average CPU usage of all the instances goes down 40 percent. Specify the following values:
 - Alarm name = `RemoveCapacity`

- Metric name = CPUUtilization
- Namespace = "AWS/EC2"
- Statistic = Average
- Period = 120
- Threshold = 40
- Comparison operator = LessThanOrEqualToThreshold
- Dimensions = "AutoScalingGroupName=my-test-asg"
- Evaluation periods = 2
- Alarm action = arn:aws:autoscaling:us-east-1:123456789012:scaling-Policy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/ my-test-asg:policyName/my-scalein-policy

For more information about the options used, see [mon-put-metric-alarm](#) in the *Amazon CloudWatch Developer Guide*.

Your command should look like the following example:

```
mon-put-metric-alarm --alarm-name RemoveCapacity --metric-name CPUUtilization
--namespace AWS/EC2
--statistic Average --period 120 --threshold 40 --comparison-operator
LessThanOrEqualToThreshold --dimensions AutoScalingGroupName=my-test-asg
--evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:123456789012:scalingPolicy:4ee9e543-86b5-4121-
b53b-aa4c23b5bbcc:autoScalingGroupName/ my-test-asg:policyName/my-scalein-
policy
```

If your request was successful, you should get a confirmation that looks like the following example:

```
OK-Created Alarm
```

Verify Your Scaling Policies and CloudWatch Alarms

To verify your CloudWatch alarms

1. Use the CloudWatch command `mon-describe-alarms` as in the following example:

```
mon-describe-alarms --headers
```

2. The command returns the following:

```
ALARM      STATE ALARM_ACTIONS  NAMESPACE  METRIC_NAME  PERIOD  STATISTIC
EVAL_PERIODS  COMPARISON    THRESHOLD
RemoveCapacity OK  arn:aws:autoscaling...policyName/my-scalein-policy AWS/EC2
CPUUtilization 120 Average 5 LessThanOrEqualToThreshold 2
AddCapacity OK  arn:aws:autoscaling...policyName/my-scaleout-policy AWS/EC2
CPUUtilization 120 Average 5 GreaterThanOrEqualToThreshold 2
```

To verify your scaling policies

1. Enter the Auto Scaling command `as-describe-policies` as in the following example:

```
as-describe-policies --auto-scaling-group my-test-asg --headers
```

2. The command returns the following:

```
SCALING-POLICY  GROUP-NAME  POLICY-NAME      SCALING-ADJUSTMENT  ADJUSTMENT-
TYPE           POLICY-ARN
SCALING-POLICY  my-test-asg      my-scalein-policy  -2                  ChangeInCapacity
arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:12bd92cc-
3597-4da5-a9c5-d06e5a076a29:autoScalingGroupName/my-test-asg:policyName/my-
scalein-policy
ALARM           ALARM-NAME      POLICY-NAME
ALARM           RemoveCapacity  my-scalein-policy
SCALING-POLICY  my-test-asg      my-scaleout-policy  30                  PercentChangeIn
Capacity arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:0331a570-
731f-4831-bc2c-12953cd9c5c6:autoScalingGroupName/my-test-asg:policyName/my-
scaleout-policy
ALARM           ALARM-NAME      POLICY-NAME
ALARM           AddCapacity     my-scaleout-policy
```

You've successfully created scaling policies and CloudWatch alarms, and you have associated your scaling policies with the CloudWatch alarms.

Scaling with Metrics Using the Query API

Use the following Auto Scaling actions to create a launch configuration, an Auto Scaling group and Auto Scaling policies.

Command	Description
CreateLaunchConfiguration	Creates a new launch configuration with specified attributes.
CreateAutoScalingGroup	Creates a new Auto Scaling group with a specified name and other attributes.
DescribeAutoScalingGroups	Describes the Auto Scaling groups, if the groups exist.
PutScalingPolicy	Creates or updates Auto Scaling policy.
DescribePolicies	Describes the policies associated with your account.

For more information about Auto Scaling actions, see Auto Scaling API Reference.

Use the following CloudWatch actions to create and describe CloudWatch alarms:

Command	Description
PutMetricAlarm	Creates an alarm and associates it with the specified CloudWatch metric and the Auto Scaling policy.

Command	Description
DescribeAlarms	Retrieves the details of the specified alarm. If no alarm is specified, retrieves details of all the alarms for the AWS account.

For more information about CloudWatch actions, see the [Amazon CloudWatch API Reference](#).

Topics

- [Create a Launch Configuration \(p. 71\)](#)
- [Create an Auto Scaling Group \(p. 71\)](#)
- [Verify Your Auto Scaling Group \(p. 72\)](#)
- [Create Scaling Policies \(p. 73\)](#)
- [Create CloudWatch Alarms \(p. 74\)](#)
- [Verify Your Scaling Policies and CloudWatch Alarms \(p. 75\)](#)

Create a Launch Configuration

Call the `CreateLaunchConfiguration` action by specifying the following parameters:

- `LaunchConfigurationName` = `my-test-lc`
- `ImageId` = `ami-514ac838`

If you don't have an AMI, and you want to find a suitable one, follow the instructions in [Finding a Suitable AMI](#).

- `InstanceType` = `m1.small`

If your request is successful, you should get a confirmation like the following example:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7c6e177f-f082-11e1-ac58-3714bEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

Create an Auto Scaling Group

Call the `CreateAutoScalingGroup` action by specifying the following parameters:

- `AutoScalingGroupName` = `my-test-asg`
- `LaunchConfigurationName` = `my-test-lc`
- `AvailabilityZones.member.1` = `us-east-1e`
- `MaxSize` = 5
- `MinSize` = 1

If your request is successful, you should get a confirmation like the following example:

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
01-01/">
  <ResponseMetadata>
```

```
<RequestId>8d798a29-f083-11e1-bdfb-cb223EXAMPLE</RequestId>
</ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

Verify Your Auto Scaling Group

Call the `DescribeAutoScalingGroups` action by specifying the following parameter.

- `AutoScalingGroupName` = `my-test-asg`

The response includes details about the group and instances launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-22T23:58:48.718Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-98e204e8</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:auto
ScalingGroup:66be2dec-ee0f-4178-8a3a-e13d91c4eba9:autoScalingGroupName/my-test-
asg<
        </AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>cb35382a-64ef-11e2-a7f1-9f203EXAMPLE</RequestId>
```

```
</ResponseMetadata>  
</DescribeAutoScalingGroupsResponse>
```

You can see that Auto Scaling launched an instance using the `my-test-lc` launch configuration; it is running (`InService`) and is healthy.

Create Scaling Policies

A scaling policy tells the Auto Scaling group what to do when the specified conditions change.

In this procedure, you'll create two scaling policies - `my-scaleout-policy` to increase the capacity of the `my-test-asg` group by 30 percent of its size, and `my-scalein-policy` to decrease the capacity of `my-test-asg` group to two instances.

To create scaling policies

1. Call `PutScalingPolicy` action by specifying the following parameters:

- `PolicyName` = `my-scaleout-policy`
- `AutoScalingGroupName` = `my-test-asg`
- `ScalingAdjustment` = 30
- `AdjustmentType` = `PercentChangeInCapacity`

Note

No Auto Scaling name, including policy names, can contain the colon (:) character because colons serve as delimiters in Amazon Resource Names (ARNs).

2. The response includes the ARN that, serves as a unique name for the new policy. Subsequently, you can use either the ARN or a combination of the policy name and group name to specify the policy.

```
<PutScalingPolicyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">  
  <PutScalingPolicyResult>  
    <PolicyARN>arn:aws:autoscaling:us-east-1:123456789012:scaling  
Policy:c7a27f55-d35e-4153-b044-8ca9155fc467:autoScalingGroupName/my-test-  
asg:policyName/my-sca  
leout-policy</PolicyARN>  
  </PutScalingPolicyResult>  
  <ResponseMetadata>  
    <RequestId>2b8415c9-657d-11e2-b53e-5db54EXAMPLE</RequestId>  
  </ResponseMetadata>  
</PutScalingPolicyResponse>
```

Copy the ARN in a safe place. You'll need it to create CloudWatch alarms.

3. Call `PutScalingPolicy` action to create another policy by specifying the following parameters:

- `PolicyName` = `my-scalein-policy`
- `AutoScalingGroupName` = `my-test-asg`
- `ScalingAdjustment` = -2
- `AdjustmentType` = `ChangeInCapacity`

4. The response includes the ARN for the policy.

```
<PutScalingPolicyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutScalingPolicyResult>
    <PolicyARN>arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:4b0551e5
-c80d-416e-a261-12140014fcef:autoScalingGroupName/my-test-asg1:policyName/my-
scalein-policy</PolicyARN>
  </PutScalingPolicyResult>
  <ResponseMetadata>
    <RequestId>5e2ab288-657d-11e2-9297-fdf721EXAMPLE</RequestId>
  </ResponseMetadata>
</PutScalingPolicyResponse>
```

Copy the ARN in a safe place. You'll need it to create CloudWatch alarms.

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 the initiation of a scaling activity based on stale metrics. By default, Auto Scaling uses a default cooldown period associated with your Auto Scaling group. To use a different cooldown period than the default specified in the Auto Scaling group, use the `Cooldown = cooldown value` parameter. 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 \(p. 46\)](#).

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 specified conditions 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.

To create CloudWatch alarms

1. Call the CloudWatch action `PutMetricAlarm` to create an alarm to increase the size of the Auto Scaling group when the average CPU usage of all the instances goes up to 80 percent by specifying the following parameters:
 - `AlarmName` = `AddCapacity`
 - `MetricName` = `CPUUtilization`
 - `Namespace` = `AWS/EC2`
 - `Statistic` = `Average`
 - `Period` = `120`
 - `Threshold` = `80`
 - `ComparisonOperator` = `GreaterThanOrEqualToThreshold`
 - `Dimensions.member.1` = `"AutoScalingGroupName=my-test-asg"`
 - `EvaluationPeriods` = `2`
 - `AlarmActions.member.1` = `arn:aws:autoscaling:us-east-1:123456789012:scaling-Policy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/ my-test-asg:policyName/my-scaleout-policy`

If your request was successful, you should get a confirmation that looks like the following example:

```
<PutMetricAlarmResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-1/">
  <ResponseMetadata>
    <RequestId>b9b6a9a8-6593-11e2-9183-b79f7EXAMPLE</RequestId>
  </ResponseMetadata>
</PutMetricAlarmResponse>
```

2. Call the CloudWatch action `PutMetricAlarm` to create an alarm to decrease the size of the Auto Scaling group when the average CPU usage of all the instances goes down to 40 percent by specifying the following parameters:
 - `AlarmName` = `RemoveCapacity`
 - `MetricName` = `CPUUtilization`
 - `Namespace` = `AWS/EC2`
 - `Statistic` = `Average`
 - `Period` = `120`
 - `Threshold` = `40`
 - `ComparisonOperator` = `LessThanOrEqualToThreshold`
 - `Dimensions.member.1` = `AutoScalingGroupName=my-test-asg`
 - `EvaluationPeriods` = `2`
 - `AlarmActions.member.1` = `arn:aws:autoscaling:us-east-1:123456789012:scaling-Policy:4ee9e543-86b5-4121- b53b-aa4c23b5bbcc:autoScalingGroupName/ my-test-asg:policyName/my-scalein-policy`

If your request was successful, you should get a confirmation that looks like the following example:

```
<PutMetricAlarmResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-1/">
  <ResponseMetadata>
    <RequestId>444dcac2-6594-11e2-9923-e57bdEXAMPLE</RequestId>
  </ResponseMetadata>
</PutMetricAlarmResponse>
```

Verify Your Scaling Policies and CloudWatch Alarms

To verify your CloudWatch alarms

1. Call the CloudWatch action `DescribeAlarms` using the following parameters:
 - `AlarmNames.member.1` = `AddCapacity`
 - `AlarmNames.member.2` = `RemoveCapacity`
2. The response includes the details about the alarms you just created. The information you get should be similar to the following example:

```
<DescribeAlarmsResult>
  <MetricAlarms>
    <member>
      .....
    </member>
  </MetricAlarms>
</DescribeAlarmsResult>
```



```

        <AlarmArn>arn:aws:cloudwatch:us-east-1:123456789012:alarm:AddCapa
city</AlarmArn>
        <AlarmConfigurationUpdatedTimestamp>2013-01-
23T17:09:57.727Z</AlarmConfigurationUpdatedTimestamp>
        <AlarmName>AddCapacity</AlarmName>
        <StateValue>OK</StateValue>
        <Period>120</Period>
        <OKActions/>
        <ActionsEnabled>true</ActionsEnabled>
        <Namespace>AWS/EC2</Namespace>
        <Threshold>80.0</Threshold>
        <EvaluationPeriods>2</EvaluationPeriods>
        <Statistic>Average</Statistic>
        <AlarmActions>
            <member>arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:c7a27f55</member>
        </AlarmActions>
        .....
        <Dimensions>
            <member>
                <Name>AutoScalingGroupName</Name>
                <Value>my-test-asg</Value>
            </member>
        </Dimensions>
        <ComparisonOperator>GreaterThanOrEqualToThreshold</ComparisonOperator>

        <MetricName>CPUUtilization</MetricName>
    </member>
    .....
    <AlarmArn>arn:aws:cloudwatch:us-east-1:123456789012:alarm:RemoveCa
pacity</AlarmArn>
    <AlarmConfigurationUpdatedTimestamp>2013-01-
23T17:13:12.560Z</AlarmConfigurationUpdatedTimestamp>
    <AlarmName>RemoveCapacity</AlarmName>
    <StateValue>ALARM</StateValue>
    <Period>120</Period>
    <OKActions/>
    <ActionsEnabled>true</ActionsEnabled>
    <Namespace>AWS/EC2</Namespace>
    <Threshold>40.0</Threshold>
    <EvaluationPeriods>2</EvaluationPeriods>
    <Statistic>Average</Statistic>
    <AlarmActions>
        <member>arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:4b0551e5</member>
    </AlarmActions>
    .....
    <Dimensions>
        <member>
            <Name>AutoScalingGroupName</Name>
            <Value>my-test-asg</Value>
        </member>
    </Dimensions>
    <ComparisonOperator>LessThanOrEqualToThreshold</ComparisonOperator>

    <MetricName>CPUUtilization</MetricName>
</member>
</MetricAlarms>

```

```
</DescribeAlarmsResult>
<ResponseMetadata>
  <RequestId>b3592828-6580-11e2-b12d-2942f7EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAlarmsResponse>
```

To verify your scaling policies

1. Call the Auto Scaling action `DescribePolicies` with the following parameter:
 - `AutoScalingGroupName = my-test-asg`
2. The response includes the details about the policies you just created. The information you get should be similar to the following example::

```
<DescribePoliciesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribePoliciesResult>
    <ScalingPolicies>
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:4b05
51e5-c80d-416e-a261-12140014fcef:autoScalingGroupName/my-test-asg:policy
Name/my-scalein-policy</PolicyARN>
        <AdjustmentType>ChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>-2</ScalingAdjustment>
        <PolicyName>my-scalein-policy</PolicyName>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Alarms/>
      </member>
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:123456789012:scaling
Policy:c7a27f55-d35e-4153-b044-8ca9155fc467:autoScalingGroupName/my-test-
asg:policyName/my
-scaleout-policy</PolicyARN>
        <AdjustmentType>PercentChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>30</ScalingAdjustment>
        <PolicyName>my-scaleout-policy</PolicyName>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Alarms/>
      </member>
    </ScalingPolicies>
  </DescribePoliciesResult>
  <ResponseMetadata>
    <RequestId>0e375e9a-658d-11e2-8ef9-07e59EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribePoliciesResponse>
```

You've successfully created scaling policies, CloudWatch alarms, and have associated your scaling policies with the CloudWatch alarms.

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.

Topics

- [Programming Considerations for Scheduled Actions \(p. 78\)](#)
- [Create Scheduled Actions \(p. 78\)](#)

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.

Create Scheduled Actions

The following steps outline how to create a scheduled action to scale your Auto Scaling group.

1. Create a launch configuration
2. Create an Auto Scaling group
3. Create a scheduled action to scale one time or on a recurring schedule
4. Verify that your Auto Scaling group is scheduled for scaling

You can create scheduled actions using the Auto Scaling command line interface (CLI) or the Query API. If you are planning to use the CLI, be sure you have installed it. For more information, see [Install the Auto Scaling CLI \(p. 16\)](#).

Topics

- [Scheduling Scaling Using the Command Line Interface \(p. 79\)](#)
- [Scheduling Scaling Using the Query API \(p. 83\)](#)

Scheduling Scaling Using the Command Line Interface

Use the following Auto Scaling commands to create a launch configuration, create and describe Auto Scaling groups, and create and describe scheduled action.

Command	Description
<code>as-create-launch-config</code>	Creates a new launch configuration with specified attributes.
<code>as-create-auto-scaling-group</code>	Creates a new Auto Scaling group with a specified name and other attributes.
<code>as-describe-auto-scaling-groups</code>	Describes the Auto Scaling groups, if the groups exist.
<code>as-put-scheduled-update-group-action</code>	Creates a scheduled action for an Auto Scaling group.
<code>as-describe-scheduled-actions</code>	Lists all the actions scheduled for your Auto Scaling group that have not been executed.

For more information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the [Auto Scaling Quick Reference Card](#).

Topics

- [Create a Launch Configuration \(p. 79\)](#)
- [Create an Auto Scaling Group \(p. 80\)](#)
- [Verify Your Auto Scaling Group \(p. 80\)](#)
- [Create a Schedule for Scaling Actions \(p. 80\)](#)
- [Verify That the Auto Scaling Group is Scheduled for Scaling \(p. 82\)](#)

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the [Basic Scenario in Auto Scaling](#).

For this procedure, specify the following values for the `as-create-launch-config` command:

- Launch configuration name = `my-test-lc`
- Image ID = `ami-514ac838`

If you don't have an AMI, and you want to find a suitable one, follow the instructions in [Finding a Suitable AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

- Instance type = `m1.small`

Your command should look similar to the following example:

```
as-create-launch-config my-test-lc --image-id ami-514ac838 --instance-type m1.small
```

If your request is successful, you should get a confirmation similar to the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Create your Auto Scaling group by using the `as-create-auto-scaling-group` command by specifying the following values:

- Auto Scaling group name = `my-test-asg`
- Launch configuration name = `my-test-lc`
- Availability Zone = `us-east-1e`
- Max size = 5
- Min size = 1

Your command should look similar to the following example:

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

If your request was successful, you should get a confirmation similar to the following example:

```
OK-Created AutoScalingGroup
```

Verify Your Auto Scaling Group

Use the `as-describe-auto-scaling-groups` command to verify your Auto Scaling group.

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg --headers
```

Note

Specify the `--headers` general option to show column headers that organize the `as-describe-auto-scaling-groups` command information.

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

```
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG  AVAILABILITY-ZONES  MIN-SIZE
MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  my-test-asg  my-test-lc      us-east-1e          1
5                1                Default
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE  STATUS  LAUNCH-CONFIG
INSTANCE  i-cbd7caba   us-east-1e        InService  InService  my-test-lc
```

You can see that Auto Scaling launched an instance using the `my-test-lc`; it is running (`InService`) and is healthy.

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

1. Use the `as-put-scheduled-update-group-action` with the following values:

- Name of your scheduled action = ScaleOut
- Auto Scaling group name = my-test-asg
- Desired Capacity = 3
- Start time = "2013-05-12T08:00:00Z"

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. Auto Scaling supports the date and time expressed in "YYYY-MM-DDThh:mm:ssZ" format in UTC/GMT only.

Your command should look similar to the following example:

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

2. You should get a confirmation similar to the following example:

```
OK-Put Scheduled Update Group Action
```

3. You might want to reduce the number of running instances in your Auto Scaling group to 1 after a certain time. Use `as-put-scheduled-update-group-action` by specifying a later date for changing the desired capacity to 1.

- Name of your scheduled action = ScaleIn
- Auto Scaling group name = my-test-asg
- Desired Capacity = 1
- Start time = "2013-05-13T08:00:00Z"

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. Auto Scaling supports the date and time expressed in "YYYY-MM-DDThh:mm:ssZ" format in UTC/GMT only.

Your command should look similar to the following example:

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

4. If your request was successful, you should get a confirmation similar to the following example:

```
OK-Put Scheduled Update Group Action
```

To create scheduled actions for scaling on a recurring schedule

1. The following example creates a scheduled action to scale on a recurring schedule that is scheduled to execute at 00:30 hours on the first of January, June, and December every year. To create a scheduled action based on a recurring schedule, use the `as-put-scheduled-update-group-action` with the following parameters:

- Name of your scheduled action = Scaleout-schedule-year

- Auto Scaling group name = my-test-asg
- Desired Capacity = 3
- Recurrence = "30 0 1 1,6,12 0"

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. You must specify the `recurrence` schedule using the Unix cron syntax format. For information about cron syntax, go to [Wikipedia](#).

Your command should look similar to the following example:

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

2. If your request was successful, you should get a confirmation similar to the following example:

```
OK-Put Scheduled Update Group Action
```

Verify That the Auto Scaling Group is Scheduled for Scaling

To verify that your Auto Scaling group is scheduled for scaling

1. Use the `as-describe-scheduled-actions` command to list all the scheduled actions attached to your Auto Scaling groups that are still waiting to be executed. After a scheduled action is completed, it is automatically deleted and no longer visible in the list of planned actions.

Your command should be similar to the following example:

```
as-describe-scheduled-actions --auto-scaling-group my-test-asg --headers
```

2. The information you get should be similar to the following example:

```
UPDATE-GROUP-ACTION my-test-asg ScaleOut 2013-05-12T08:00:00Z 3  
UPDATE-GROUP-ACTION my-test-asg ScaleIn 2013-01-13T08:00:00Z 1
```

3. If the scheduled action is already executed, use `as-describe-scaling-activities` by specifying the name of your Auto Scaling group and using the `--show-xml` option.

Your command should look similar to the following example:

```
as-describe-scaling-activities --auto-scaling-group my-test-asg --show-  
xml
```

4. The information you get should be similar to the following example:

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon  
aws.com/doc/2  
011-01-01/">  
  <DescribeScalingActivitiesResult>  
    <NextToken>71382dd3-75af-4a57-9c23-988fdcb1866a</NextToken>
```

```
<Activities>
  <member>
    <StatusCode>Successful</StatusCode>
    <Progress>100</Progress>
    <ActivityId>7d1a9dd2-7b53-4334-8a5f-5fa2a9731d64</ActivityId>
    <StartTime>2013-01-30T03:00:51.931Z</StartTime>
    <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
    <Cause>At 2013-01-30T03:00:21Z a scheduled action update of AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 1 to 3. At 2013-01-30T03:00:21Z the scheduled action ScaleOut executed. Setting desired capacity from 1 to 3. At 2013-01-30T03:00:51Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 3.</Cause>
    <Details>{}</Details>
    <Description>Launching a new EC2 instance: i-aacc62da</Description>

    <EndTime>2013-01-30T03:02:01Z</EndTime>
  </member>
```

You can determine whether instances were launched due to a scheduled action by examining the description in the `Cause` field. Activities launched as a direct result of a scheduled action have a reference to the specific action name in the `Cause` field of the corresponding scaling activity. The `Cause` field in the example shows the scheduled action taken for the Auto Scaling group `my-test-asg`.

You have successfully created a scaling plan for your Auto Scaling group based either on a specific time or recurring schedule.

Scheduling Scaling Using the Query API

Use the following Auto Scaling actions to create a launch configuration, create and describe Auto Scaling groups, and create and describe a scheduled action.

Command	Description
CreateLaunchConfiguration	Creates a new launch configuration with specified attributes.
CreateAutoScalingGroup	Creates a new Auto Scaling group with a specified name and other attributes.
DescribeAutoScalingGroups	Describes the Auto Scaling groups, if the groups exist.
PutScheduledUpdateGroupAction	Creates a scheduled action for an Auto Scaling group..
DescribeScheduledActions	Lists all the actions scheduled for your Auto Scaling group that have not been executed.

For more information about Auto Scaling actions, see the [Auto Scaling API Reference](#).

Topics

- [Create a Launch Configuration \(p. 84\)](#)
- [Create an Auto Scaling Group \(p. 84\)](#)
- [Verify Your Auto Scaling Group \(p. 84\)](#)

- [Create a Schedule for Scaling Actions](#) (p. 85)
- [Verify That Your Auto Scaling Group is Scheduled for Scaling](#) (p. 87)

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the [Basic Scenario in Auto Scaling](#).

For this procedure, specify the following parameters for the `CreateLaunchConfiguration` action:

- `LaunchConfigurationName` = `my-test-lc`
- `ImageId` = `ami-514ac838`

If you don't have an AMI, and you want to find a suitable one, follow the instructions in [Finding a Suitable AMI](#).

- `InstanceType` = `m1.small`

If your request is successful, you should get a confirmation similar to the following example:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7c6e177f-f082-11e1-ac58-3714bEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

Create an Auto Scaling Group

Call the `CreateAutoScalingGroup` action to create your Auto Scaling group by specifying the following parameters:

- `AutoScalingGroupName` = `my-test-asg`
- `LaunchConfigurationName` = `my-test-lc`
- `AvailabilityZone.member.1` = `us-east-1e`
- `MaxSize` = 5
- `MinSize` = 1

If your request is successful, you should get confirmation similar to the following example:

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
01-01/">
  <ResponseMetadata>
    <RequestId>8d798a29-f083-11e1-bdfb-cb223EXAMPLE</RequestId>
  </ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

Verify Your Auto Scaling Group

Call the `DescribeAutoScalingGroups` action with the following parameters to verify your Auto Scaling group.

- `AutoScalingGroupName` = `my-test-asg`

If your request is successful, you should get a response similar to the following example. The response shows the details about the group and the instances launched.

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-22T23:58:48.718Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-98e204e8</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:autoScalingGroup:66be2dec-ee0f-4178-8a3a-e13d91c4eba9:autoScalingGroupName/my-test-asg</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>cb35382a-64ef-11e2-a7f1-9f203EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

You can see that Auto Scaling launched an instance using the `my-test-lc` launch configuration; it is running (`InService`) and it is healthy.

Create a Schedule for Scaling Actions

You can create a schedule for scaling on a specific date and time or for scaling based on a daily, weekly, monthly, or yearly schedule.

To schedule scaling on a specific date and time

1. Call the `PutScheduledUpdateGroupAction` action using the following parameters:

- `ScheduledActionName` = `ScaleOut`
- `AutoScalingGroupName` = `my-test-asg`
- `DesiredCapacity` = `3`
- `StartTime` = `"2013-05-12T08:00:00Z"`

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. Auto Scaling supports the date and time expressed in "YYYY-MM-DDThh:mm:ssZ" format in UTC/GMT only.

2. If your request is successful, you should get a confirmation similar to the following example:

```
<PutScheduledUpdateGroupActionResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>757957fc-6a5f-11e2-bb90-e9977EXAMPLE</RequestId>
  </ResponseMetadata>
</PutScheduledUpdateGroupActionResponse>
```

3. You might want to reduce the number of running instances in your Auto Scaling group to 1 after a certain time. Call the `PutScheduledUpdateGroupAction` action using the following parameters:

- `ScheduledActionName` = `ScaleIn`
- `AutoScalingGroupName` = `my-test-asg`
- `DesiredCapacity` = `1`
- `StartTime` = `"2013-05-13T08:00:00Z"`

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. Auto Scaling supports the date and time expressed in "YYYY-MM-DDThh:mm:ssZ" format in UTC/GMT only.

4. If your request is successful, you should get a confirmation similar to the following example:

```
<PutScheduledUpdateGroupActionResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>75cba157-6a60-11e2-a62c-b1235EXAMPLE</RequestId>
  </ResponseMetadata>
</PutScheduledUpdateGroupActionResponse>
```

To create scheduled actions for scaling on a recurring basis

1. The following example creates a scheduled action to scale on a recurring schedule that is scheduled to execute at 00:30 hours on the first of January, June, and December every year. To create a scheduled action based on a recurring schedule, call the `PutScheduledUpdateGroupAction` action with the following parameters:

- `ScheduledActionName = Scaleout-schedule-year`
- `AutoScalingGroupName = my-test-asg`
- `DesiredCapacity = 3`
- `Recurrence = "30 0 1 1,6,12 0"`

Note

If you try to schedule your action in the past, Auto Scaling returns an error message. You must specify a `Recurrence` schedule using the Unix cron syntax format. For information about cron syntax, go to [Wikipedia](#).

2. If your request is successful, you should get a confirmation similar to the following example:

```
<PutScheduledUpdateGroupActionResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>3bc8c9bc-6a62-11e2-8a51-4b8a1EXAMPLE</RequestId>
  </ResponseMetadata>
</PutScheduledUpdateGroupActionResponse>
```

Verify That Your Auto Scaling Group is Scheduled for Scaling

To verify that your Auto Scaling group is scheduled for scaling

1. Call the `DescribeScheduledActions` action to list all the scheduled actions attached to your Auto Scaling groups that are still waiting to be executed. After a scheduled action is completed, it is automatically deleted and no longer visible in the list of planned actions. Use the following parameter to narrow down the result to your Auto Scaling group:

- `AutoScalingGroupName = my-test-asg`

The response lists all the actions scheduled for your Auto Scaling group.

2. If the scheduled action is already executed, use `DescribeScalingActivities` by specifying the name of your Auto Scaling group.
3. If your request is successful, you should get response similar to the following example. The response shows details about all the scaling activities for your Auto Scaling group.

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2
011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>71382dd3-75af-4a57-9c23-988fdcb1866a</NextToken>
    <Activities>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>100</Progress>
        <ActivityId>7d1a9dd2-7b53-4334-8a5f-5fa2a9731d64</ActivityId>
        <StartTime>2013-01-30T03:00:51.931Z</StartTime>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Cause>At 2013-01-30T03:00:21Z a scheduled action update of AutoScal
ingGroup constraints to min: 1, max: 5, desired: 3 changing the desired ca
pacity fro
m 1 to 3. At 2013-01-30T03:00:21Z the scheduled action ScaleOut executed.
```

```
Setting desired capacity from 1 to 3. At 2013-01-30T03:00:51Z an instance
was started
in response to a difference between desired and actual capacity, increasing
the capacity from 1 to 3.</Cause>
  <Details>{}</Details>
  <Description>Launching a new EC2 instance: i-aacc62da</Description>

  <EndTime>2013-01-30T03:02:01Z</EndTime>
</member>
```

You can determine if instances were launched due to a scheduled action by examining the description in the `Cause` field. Activities launched as a direct result of a scheduled action have a reference to the specific action name in the `Cause` field of the corresponding scaling activity. The `Cause` field in the example shows the scheduled action taken for the Auto Scaling group `my-test-asg`.

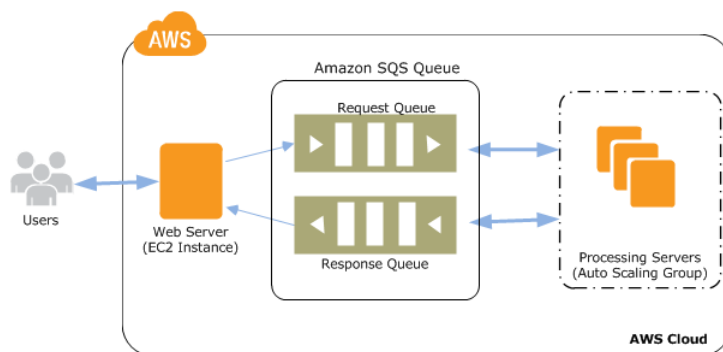
You have successfully created a scaling plan for your Auto Scaling group based on either a specific time or a recurring schedule.

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 one component in the application generates to be consumed by another component. A queue is a temporary repository for messages that are awaiting processing. When you use Amazon SQS queues in your architecture, you introduce an interface between the various components, which allows the components to be decoupled from one another. The independent components are easy to scale. This topic shows you how you can use Amazon SQS queues to establish thresholds that Auto Scaling can use to scale the components in your architecture.

For more information about Amazon SQS, see [What is Amazon Simple Queue Service?](#) in the *Amazon Simple Queue Service Developer Guide*. If you aren't already acquainted with the basic concepts behind Auto Scaling, see [What is Auto Scaling? \(p. 1\)](#).

For example, you have a web application that receives orders from customers. The application, places the orders in a queue until they are picked up for processing, processes the orders, and then sends the processed orders back to the customer. In this example, you have an application running on an EC2 instance that receives the orders and sends them to the Amazon SQS queue. The Amazon SQS queue stores the orders until they are ready to be processed. You have configured an Auto Scaling group to receive the orders from the queue, process the orders, and send them back to the queue. For this example, the capacity in your Auto Scaling group is configured to handle a normal order load. 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 you start getting more orders than you normally do, and later you would terminate those additional EC2 instances when the orders return to normal levels. If your order levels increase and decrease on a predictable schedule, you can specify the time and date to perform these scaling activities. For more information, see [Scheduled Scaling \(p. 78\)](#).

What happens if you can't predict when you will get more orders? In that case, you have to identify the conditions that determine the increasing and decreasing order loads. Then you need to tell Auto Scaling to launch or terminate EC2 instances when those conditions are met. 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. By performing load tests on the queue, you can determine the optimal length of the queue (the length at which you have the required number of EC2 instances running to cover the demand). At any time, if the current length of the queue exceeds the optimal length, then you should start additional EC2 instances. Likewise, if the current length falls below the optimal length, then it's time to terminate the additional EC2 instances.

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 sections step you through the process of creating Auto Scaling policies for configuring your Auto Scaling group to scale based on the number of messages in your Amazon SQS queue. Policies tell Auto Scaling what to do when the specified conditions occur. For information about using Auto Scaling policies, see [Dynamic Scaling \(p. 57\)](#). You can create Auto Scaling policies using the Auto Scaling command line interface (CLI) or the Query API.

Create Auto Scaling Policies and CloudWatch Alarms

Topics

- [Scaling with Amazon SQS Using the Command Line Interface \(p. 90\)](#)
- [Scaling with Amazon SQS Using the Query API \(p. 95\)](#)

The following walkthroughs will show you how to create policies for scaling in and scaling out, plus create, verify, and validate CloudWatch alarms for your scaling policies.

1. How to create an Auto Scaling policy to scale out (launch instances).
2. How to create another Auto Scaling policy to scale in (terminate instances).

3. How to create an CloudWatch alarm to watch the Amazon SQS metric `ApproximateNumberOfMessagesVisible` and then attach either the Auto Scaling policy for scaling in or for scaling out.
4. How to verify that your Auto Scaling policies and CloudWatch alarms have been created.
5. How to validate that the scaling activity is taking place when the specified condition occurs.

Prerequisites

Before you begin creating Auto Scaling policies and CloudWatch alarms, be sure you have completed the following prerequisites:

- Sign up for AWS.

If you haven't yet signed up, go to <http://aws.amazon.com>, click **Sign Up**, and follow the on-screen instructions.

- Follow the steps in [Working with Amazon SQS](#) in the *Amazon Simple Queue Service Developer Guide* to create a queue, confirm that the queue exists, and send and receive messages to and from your queue.
- Create an Auto Scaling group. For more information, see [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#). Make a note of the instance ID of the newly launched instance, as you'll need this later.
- Configure the EC2 instance in your Auto Scaling group to receive, send, and process messages using Amazon SQS.

Scaling with Amazon SQS Using the Command Line Interface

In this procedure, you'll create two scaling policies, `my-sqs-scaleout-policy`, which increases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance, and `my-sqs-scalein-policy`, which decreases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance. You can optionally 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. For more information, see [Understanding Auto Scaling Cooldowns \(p. 46\)](#).

Create Scaling Policies

Use the CLI to create policies for scaling out and scaling in.

To create scaling policies

1. Use the `as-put-scaling-policy` command and specify the following values:
 - Policy name = `as-sqs-scaleout-policy`
 - Auto Scaling group name = `my-test-asg`
 - Adjustment = `1`
 - Adjustment type = `ChangeInCapacity`
 - [optional] Cooldown = *cooldown period*

Your command should look similar to the following example:

```
as-put-scaling-policy my-sqs-scaleout-policy --auto-scaling-group my-test-asg --adjustment=1 --type ChangeInCapacity
```

Note

If you are running Windows, you must use quotation marks when specifying the adjustment value, for example `--adjustment=-1`.

No Auto Scaling name, including a policy name, can contain the colon (:) character because colons serve as delimiters in [Amazon Resource Names \(ARNs\)](#).

2. Auto Scaling returns the ARN that serves as a unique name for the new policy. Subsequently, you can use either the ARN or a combination of the policy name and group name to specify the policy.

```
arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy
```

Make a copy of the ARN, and put it in a safe place. You'll need it to create CloudWatch alarms.

3. Use the `as-put-scaling-policy` command to create another policy by specifying the following values:
 - Policy name = `as-sqs-scalein-policy`
 - Auto Scaling group name = `my-test-asg`
 - Adjustment = `-1`
 - Adjustment type = `ChangeIn Capacity`
 - [optional] Cooldown = *cooldown period*

Your command should look similar to the following example:

```
as-put-scaling-policy my-sqs-scalein-policy --auto-scaling-group my-test-asg --adjustment=-1 --type ChangeInCapacity
```

4. Auto Scaling returns the ARN for the policy.

```
arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy
```

Make a copy of the ARN, and put it in a safe place. You'll need it to create CloudWatch alarms.

Create CloudWatch Alarms

In the previous procedure, 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 occur. In this procedure, you create alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies.

Before you begin, be sure you have installed the CloudWatch CLI. For more information, see [Command Line Tools](#).

To create CloudWatch alarms

1. Use the CloudWatch command `mon-put-metric-alarm` to create an alarm to increase 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 more for a period of five minutes.

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.

Specify the following values:

- Alarm name = `AddCapacityToProcessQueue`
- Metric name = `ApproximateNumberOfMessagesVisible`
- Namespace = `"AWS/SQS"`
- Statistic = `Average`
- Period = `300`
- Threshold = `3`
- Comparison operator = `GreaterThanOrEqualToThreshold`
- Dimensions = `"QueueName=MyQueue"`
- Evaluation periods = `2`
- Alarm action = `arn:aws:autoscaling:us-east-1:803981987763:scaling-Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy`

Your command should look like the following example:

```
prompt>mon-put-metric-alarm --alarm-name AddCapacityToProcessQueue --metric-name ApproximateNumberOfMessagesVisible --namespace "AWS/SQS" --statistic Average --period 300 --threshold 3 --comparison-operator GreaterThanOrEqualToThreshold --dimensions "QueueName=MyQueue" --evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy
```

If your request was successful, you should get a confirmation that looks like the following example:

```
OK-Created Alarm
```

2. Use the CloudWatch command [mon-put-metric-alarm](#) to create an alarm to decrease the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) decreases to one and remains one or less for a period of five minutes. Specify the following values:

- Alarm name = `RemoveCapacityFromTheProcessQueue`
- Metric name = `ApproximateNumberOfMessagesVisible`
- Namespace = `"AWS/SQS"`
- Statistic = `Average`
- Period = `300`
- Threshold = `1`
- Comparison operator = `LessThanOrEqualToThreshold`
- Dimensions = `"QueueName=MyQueue"`
- Evaluation periods = `2`
- Alarm action = `arn:aws:autoscaling:us-east-1:803981987763:scaling-Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy`

Your command should look like the following example:

```
mon-put-metric-alarm --alarm-name RemoveCapacityFromTheProcessQueue --metric-  
name ApproximateNumberOfMessagesVisible --namespace "AWS/SQS"  
--statistic Average --period 300 --threshold 1 --comparison-operator  
LessThanOrEqualToThreshold --dimensions "QueueName=MyQueue"  
--evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-  
1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScal-  
ingGroupName/my-test-asg:policyName/my-sqs-scalein-policy
```

If your request was successful, you should get a confirmation that looks like the following example:

```
OK-Created Alarm
```

Verify Your Scaling Policies and CloudWatch Alarms

You can use the CLI to verify if your CloudWatch alarms and scaling policies are created.

To verify your CloudWatch alarms

1. Use the CloudWatch command `mon-describe-alarms` and specify the following values:
 - Alarm names = `AddCapacityToProcessQueue` `RemoveCapacityFromTheProcessQueue`

Your command should look like the following example:

```
mon-describe-alarms AddCapacityToProcessQueue RemoveCapacityFromThePro-  
cessQueue --headers
```

2. The command returns the following information:

```
ALARM      STATE ALARM_ACTIONS  NAMESPACE  METRIC_NAME  PERIOD  STATISTIC  
EVAL_PERIODS  COMPARISON    THRESHOLD  
RemoveCapacityFromProcessQueue OK arn:aws:autoscaling...policyName/my-sqs-  
scalein-policy AWS/SQS ApproximateNumberOfMessagesVisible 300 Average 5  
LessThanOrEqualToThreshold 1  
AddCapacityToProcessQueue OK arn:aws:autoscaling...:policyName/my-sqs-sca-  
leout-policy AWS/SQS ApproximateNumberOfMessagesVisible 300 Average 5  
GreaterThanOrEqualToThreshold 3
```

To verify your scaling policies

1. Enter the Auto Scaling command `as-describe-policies`, as in the following example:

```
as-describe-policies --auto-scaling-group my-test-asg --headers
```

2. The command returns the following information:

```

SCALING-POLICY  GROUP-NAME  POLICY-NAME      SCALING-ADJUSTMENT  ADJUSTMENT-
TYPE           POLICY-ARN
SCALING-POLICY  my-test-asg    my-sqs-scalein-policy  1    ChangeInCapacity
               arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-
77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-
sqs-scalein-policy
ALARM           ALARM-NAME      POLICY-NAME
ALARM           RemoveCapacityFromTheProcessQueue    my-sqs-scalein-policy
SCALING-POLICY  my-test-asg    my-sqs-scaleout-policy  1    ChangeInCapa
city           arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-
9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-
sqs-scaleout-policy
ALARM           ALARM-NAME      POLICY-NAME
ALARM           AddCapacityToProcessQueue    my-sqs-scaleout-policy

```

Test for Scaling Out and Scaling In

You can test if your Auto Scaling group increases its capacity (that is, it launches one or more EC2 instances) when the number of messages in your Amazon SQS queue increases. First, you'll have to increase the number of messages in your Amazon SQS queue, and then you must verify that your Auto Scaling group has launched an additional EC2 instance. Likewise, to test if your Auto Scaling group decreases when the number of messages in the Amazon SQS queue decreases, you'll have to remove messages from the queue and then verify that the Auto Scaling group terminates an EC2 instance.

To test if an EC2 instance is launched when messages in the queue increase

1. Follow the steps in [Sending a Message](#) to add messages to the Amazon SQS queue you created earlier for this example. Make sure you have at least three messages in the queue.
2. 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.
3. Use the `as-describe-auto-scaling-groups` command and specify the following value:
 - Auto Scaling group name: `my-test-asg`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg --header
```

The command returns the following information:

```

AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG  AVAILABILITY-ZONES  MIN-SIZE
MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  my-test-asg  my-test-lc      us-east-1e          1
10          1          Default
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS  LAUNCH-CONFIG
INSTANCE  i-2cd22f5c   us-east-1e        InService  Healthy my-test-lc
INSTANCE  i-5a277829   us-east-1e        InService  Healthy my-test-lc

```

Your Auto Scaling group has launched an additional EC2 instance.

To test if an EC2 instance terminates when messages in the queue decrease

1. Follow the steps in [Deleting a Message](#) to remove messages from the Amazon SQS queue. Make sure you have no more than one message in the queue.
2. 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.
3. Use `as-describe-auto-scaling-groups` command and specify the following value:
 - Auto Scaling group name: `my-test-asg`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg --header
```

The command returns the following information:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1e	1	10	1	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-5a277829	us-east-1e	InService	Healthy	my-test-lc		

Your Auto Scaling group has terminated the additional EC2 instance.

You've successfully created Auto Scaling policies and CloudWatch alarms to scale based on the number of messages in the Amazon SQS queue waiting to be processed. You have associated your scaling policies with the CloudWatch alarms, and you have tested to verify that your Auto Scaling group scales when the specified conditions occur.

Scaling with Amazon SQS Using the Query API

If you aren't already acquainted with the procedure for creating a Query API request, see [Use Query Requests to Call Auto Scaling APIs \(p. 23\)](#).

In this procedure, you'll create two scaling policies, `my-sqs-scaleout-policy`, which increases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance, and `my-sqs-scalein-policy`, which decreases the capacity of `my-test-asg` Auto Scaling group by one EC2 instance. You can optionally 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. For more information, see [Understanding Auto Scaling Cooldowns \(p. 46\)](#).

Create Scaling Policies

Use the Query API to create policies for scaling out and scaling in.

To create scaling policies

1. Call the `PutScalingPolicy` action and specify the following parameters:
 - `PolicyName` = `as-sqs-scaleout-policy`
 - `AutoScalingGroupName` = `my-test-asg`
 - `ScalingAdjustment` = `1`
 - `AdjustmentType` = `ChangeInCapacity`

- [optional] Cooldown = *cooldown period*

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&ScalingAdjustment=1
&AdjustmentType=ChangeInCapacity
&PolicyName=my-sqs-scaleout-policy
&Version=2011-01-01
&Action=PutScalingPolicy
&AUTHPARAMS
```

Note

If you are running Windows, you must use quotation marks when specifying the adjustment value, for example "--adjustment=-1".

No Auto Scaling name, including a policy name, can contain the colon (:) character because colons serve as delimiters in [Amazon Resource Names \(ARNs\)](#).

2. Auto Scaling returns the ARN that serves as a unique name for the new policy. Subsequently, you can use either the ARN or a combination of the policy name and group name to specify the policy.

```
<PutScalingPolicyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutScalingPolicyResult>
    <PolicyARN> arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy </PolicyARN>
  </PutScalingPolicyResult>
  <ResponseMetadata><RequestId>2b8415c9-657d-11e2-b53e-5db54EXAMPLE</RequestId>
</ResponseMetadata>
</PutScalingPolicyResponse>
```

Make a copy of the ARN and put it in a safe place. You'll need it to create CloudWatch alarms.

3. Call the [PutScalingPolicy](#) action to create another policy by specifying the following parameters:
 - PolicyName = as-sqs-scalein-policy
 - AutoScalingGroupName = my-test-asg
 - ScalingAdjustment = -1
 - AdjustmentType = ChangeInCapacity
 - [optional] Cooldown = *cooldown period*

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&ScalingAdjustment=-1
&AdjustmentType=ChangeInCapacity
&PolicyName=my-sqs-scalein-policy
&Version=2011-01-01
&Action=PutScalingPolicy
&AUTHPARAMS
```

4. Auto Scaling returns the ARN for the policy.

```
<PutScalingPolicyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutScalingPolicyResult>
    <PolicyARN> arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy
    </PolicyARN>
  </PutScalingPolicyResult>
  <ResponseMetadata><RequestId>2b8415c9-657d-11e2-b53e-5db54EXAMPLE</RequestId>
</ResponseMetadata>
</PutScalingPolicyResponse>
```

Make a copy of the ARN and put it in a safe place. You'll need it to create CloudWatch alarms.

Create CloudWatch Alarms

In the previous procedure, 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 occur. In this procedure, you create alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies.

To create CloudWatch alarms

1. Use the CloudWatch action [PutMetricAlarm](#) to create an alarm to increase 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 more for a period of five minutes.

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.

Specify the following values:

- `AlarmName` = `AddCapacityToProcessQueue`
- `MetricName` = `ApproximateNumberOfMessagesVisible`
- `Namespace` = `"AWS/SQS"`
- `Statistic` = `Average`
- `Period` = `300`
- `Threshold` = `3`
- `ComparisonOperator` = `GreaterThanOrEqualToThreshold`
- `Dimensions.member.1` = `"QueueName=MyQueue"`
- `EvaluationPeriods` = `2`
- `AlarmActions.member.1` = `arn:aws:autoscaling:us-east-1:803981987763:scaling-Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy`

Your request should look like the following example:

```
https://monitoring.us-east-1.amazonaws.com/?Statistic=Average
&Threshold=3
&EvaluationPeriods=2
```

```
&ComparisonOperator=GreaterThanOrEqualToThreshold
&AlarmName=AddCapacityToProcessQueue
&AlarmActions.member.1= arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scaleout-policy
&Namespace=AWS/SQS
&Dimensions.member.1.Name=QueueName
&Dimensions.member.1.Value=MyQueue
&Period=300
&MetricName=ApproximateNumberOfMessagesVisible
&Version=2010-08-01
&Action=PutMetricAlarm
&AUTHPARAMS
```

If your request was successful, you should get a confirmation that looks like the following example:

```
<PutMetricAlarmResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-
1/">
  <ResponseMetadata>
    <RequestId>b9b6a9a8-6593-11e2-9183-b79f7EXAMPLE</RequestId>
  </ResponseMetadata>
</PutMetricAlarmResponse>
```

2. Use the CloudWatch action [PutMetricAlarm](#) to create an alarm to decrease the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) decreases to one and remains one or less for a period of five minutes. Specify the following parameters:
 - `AlarmName` = `RemoveCapacityFromTheProcessQueue`
 - `MetricName` = `ApproximateNumberOfMessagesVisible`
 - `Namespace` = `"AWS/SQS"`
 - `Statistic` = `Average`
 - `Period` = `300`
 - `Threshold` = `1`
 - `ComparisonOperator` = `LessThanOrEqualToThreshold`
 - `Dimensions.member.1` = `"QueueName=MyQueue"`
 - `EvaluationPeriods` = `2`
 - `AlarmActions.member.1` = `arn:aws:autoscaling:us-east-1:803981987763:scaling-Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy`

Your request should look like the following example:

```
https://monitoring.us-east-1.amazonaws.com/?Statistic=Average
&Threshold=1
&EvaluationPeriods=2
&ComparisonOperator=LessThanOrEqualToThreshold
&AlarmName=RemoveCapacityFromTheProcessQueue
&AlarmActions.member.1= arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scalein-policy
&Namespace=AWS/SQS
```

```
&Dimensions.member.1.Name=QueueName
&Dimensions.member.1.Value=MyQueue
&Period=300
&MetricName=ApproximateNumberOfMessagesVisible
&Version=2010-08-01
&Action=PutMetricAlarm
&AUTHPARAM
```

If your request was successful, you should get a confirmation that looks like the following example:

```
<PutMetricAlarmResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-01/">
  <ResponseMetadata>
    <RequestId>b9b6a9a8-6593-11e2-9183-b79f7EXAMPLE</RequestId>
  </ResponseMetadata>
</PutMetricAlarmResponse>
```

Verify Your Scaling Policies and CloudWatch Alarms

You can use the Query API to verify if your CloudWatch alarms and scaling policies are created.

To verify your CloudWatch alarms

1. Call the CloudWatch action [DescribeAlarms](#), and specify the following parameters:

- AlarmNames.member.1 = AddCapacityToProcessQueue
- AlarmNames.member.2 = RemoveCapacityFromTheProcessQueue

Your request should look like the following example:

```
https://monitoring.us-east-1.amazonaws.com/?AlarmNames.member.1=AddCapacityToProcessQueue
&AlarmNames.member.2=RemoveCapacityFromTheProcessQueue
&MaxRecords=50
&Version=2010-08-01
&Action=DescribeAlarms
&AUTHPARAMS
```

2. The response includes the details about the alarms you just created. The information you get should be similar to the following example:

```
<DescribeAlarmsResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-01/">
  <DescribeAlarmsResult>
    <MetricAlarms>
      <member>
        <StateUpdatedTimestamp>2013-02-25T21:36:22.428Z</StateUpdatedTimestamp>
        <InsufficientDataActions/>
        <AlarmArn>arn:aws:cloudwatch:us-east-1:803981987763:alarm:AddCapacityToProcessQueue</AlarmArn>
        <AlarmConfigurationUpdatedTimestamp>2013-02-
```



```
25T21:36:22.428Z</AlarmConfigurationUpdatedTimestamp>
  <AlarmName>AddCapacityToProcessQueue</AlarmName>
  <StateValue>INSUFFICIENT_DATA</StateValue>
  <Period>300</Period>
  <OKActions/>
  <ActionsEnabled>true</ActionsEnabled>
  <Namespace>AWS/SQS</Namespace>
  <Threshold>3.0</Threshold>
  <EvaluationPeriods>2</EvaluationPeriods>
  <Statistic>Average</Statistic>
  <AlarmActions>
    <member>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scaleout-policy</member>
  </AlarmActions>
  <StateReason>Unchecked: Initial alarm creation</StateReason>
  <Dimensions>
    <member>
      <Name>AutoScalingGroupName</Name>
      <Value>my-test-asg</Value>
    </member>
  </Dimensions>
  <ComparisonOperator>GreaterThanOrEqualToThreshold</ComparisonOperator>

  <MetricName>ApproximateNumberOfMessagesVisible</MetricName>
</member>
<member>
  <StateUpdatedTimestamp>2013-02-25T23:28:12.092Z</StateUpdated
Timestamp>
  <InsufficientDataActions/>
  <AlarmArn>arn:aws:cloudwatch:us-east-1:803981987763:alarm:RemoveCa
pacityFromProcessQueue</AlarmArn>
  <AlarmConfigurationUpdatedTimestamp>2013-02-
25T23:28:12.092Z</AlarmConfigurationUpdatedTimestamp>
  <AlarmName>RemoveCapacityFromProcessTheQueue</AlarmName>
  <StateValue>INSUFFICIENT_DATA</StateValue>
  <Period>300</Period>
  <OKActions/>
  <ActionsEnabled>true</ActionsEnabled>
  <Namespace>AWS/SQS</Namespace>
  <Threshold>1.0</Threshold>
  <EvaluationPeriods>2</EvaluationPeriods>
  <Statistic>Average</Statistic>
  <AlarmActions>
    <member>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scalein-policy</member>
  </AlarmActions>
  <StateReason>Unchecked: Initial alarm creation</StateReason>
  <Dimensions>
    <member>
      <Name>QueueName</Name>
      <Value>MyQueue</Value>
    </member>
  </Dimensions>
  <ComparisonOperator>LessThanOrEqualToThreshold</ComparisonOperator>

  <MetricName>ApproximateNumberOfMessagesVisible</MetricName>
```

```
</member>
</MetricAlarms>
</DescribeAlarmsResult>
<ResponseMetadata>
  <RequestId>cc132626-7fa3-11e2-9059-ebdEXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAlarmsResponse>
```

To verify your scaling policies

1. Call the Auto Scaling action [DescribePolicies](#) by specifying the following parameter:

- AutoScalingGroupName = my-test-asg

Your request should look like the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&MaxRecords=20
&Version=2011-01-01
&Action=DescribePolicies
&AUTHPARAMS
```

2. The response includes the details about the policies you just created. The information you get should be similar to the following example:

```
<DescribePoliciesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribePoliciesResult>
    <ScalingPolicies>
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-
sqs-scalein-policy</PolicyARN>
        <AdjustmentType>ChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>1</ScalingAdjustment>
        <PolicyName>my-sqs-scalein-policy</PolicyName>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Alarms>
          <member>
            <AlarmName>RemoveCapacityFromTheProcessQueue</AlarmName>
            <AlarmARN>arn:aws:cloudwatch:us-east-1:803981987763:alarm:Remove
CapacityFromProcessQueue</AlarmARN>
          </member>
        </Alarms>
      </member>
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-
sqs-scaleout-policy</PolicyARN>
        <AdjustmentType>ChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>1</ScalingAdjustment>
```

```
<PolicyName>my-sqs-scaleout-policy</PolicyName>
<AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
<Alarms>
  <member>
    <AlarmName>AddCapacityToProcessQueue</AlarmName>
    <AlarmARN>arn:aws:cloudwatch:us-east-1:803981987763:alarm:AddCa
capacityToProcessQueue</AlarmARN>
  </member>
</Alarms>
</member>
</ScalingPolicies>
</DescribePoliciesResult>
<ResponseMetadata>
  <RequestId>01759ddc-7fa6-11e2-8dbf-5fe8fEXAMPLE</RequestId>
</ResponseMetadata>
</DescribePoliciesResponse>
```

Test for Scaling Out and Scaling In

You can test if your Auto Scaling group increases its capacity (that is, it launches one or more EC2 instances) when the number of messages in your Amazon SQS queue increases. First, you have to increase the number of messages in your Amazon SQS queue, and then you must verify that your Auto Scaling group has launched an additional EC2 instance. Likewise, to test if your Auto Scaling group decreases when the number of messages in the Amazon SQS queue decreases, you have to remove messages from the queue and then verify that the Auto Scaling group terminates an EC2 instance.

To test if an EC2 instance is launched when messages in the queue increase

1. Follow the steps in [Sending a Message](#) to add messages in the Amazon SQS queue you created earlier for this example. Make sure you have at least three messages in the queue.
2. 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.
3. Call the [DescribeAutoScalingGroups](#) action by specifying the following parameter:
 - Auto Scaling group name: `my-test-asg`

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-
asg
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The response includes details about the group and instances launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResult>
  <AutoScalingGroups>
    <member>
      <Tags/>
      <SuspendedProcesses/>
```

```
<AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
<HealthCheckType>EC2</HealthCheckType>
<CreatedTime>2013-01-21T23:06:56.574Z</CreatedTime>
<EnabledMetrics/>
<LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
<Instances>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1e</AvailabilityZone>
    <InstanceId>i-2cd22f5c</InstanceId>
    <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
    <LifecycleState>InService</LifecycleState>
  </member>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1e</AvailabilityZone>
    <InstanceId>i-5a277829</InstanceId>
    <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
    <LifecycleState>InService</LifecycleState>
  </member>
</Instances>
<DesiredCapacity>1</DesiredCapacity>
<AvailabilityZones>
  <member>us-east-1e</member>
</AvailabilityZones>
<LoadBalancerNames/>
<MinSize>1</MinSize>
<VPCZoneIdentifier/>
<HealthCheckGracePeriod>0</HealthCheckGracePeriod>
<DefaultCooldown>300</DefaultCooldown>
<AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:bf6a47f1-1eda-4108-bf48-8d8da0c2b72e:auto
ScalingGroupName/my-test-asg</
AutoScalingGroupARN>
  <TerminationPolicies>
    <member>Default</member>
  </TerminationPolicies>
  <MaxSize>10</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>7e20e72f-851c-11e2-b688-ff8f1EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Take a look at the `member` fields. Your Auto Scaling group has launched an additional EC2 instance.

To test if the EC2 instance in your Auto Scaling group terminates when messages in the queue decrease

1. Follow the steps in [Deleting a Message](#) to remove messages from the Amazon SQS queue. Make sure you have no more than one message in the queue.
2. 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.

3. Call the [DescribeAutoScalingGroups](#) action by specifying the following parameter:

- Auto Scaling group name: my-test-asg

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-
asg
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The response includes details about the group and instances launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResult>
  <AutoScalingGroups>
    <member>
      <Tags/>
      <SuspendedProcesses/>
      <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
      <HealthCheckType>EC2</HealthCheckType>
      <CreatedTime>2013-01-21T23:06:56.574Z</CreatedTime>
      <EnabledMetrics/>
      <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
      <Instances>
        <member>
          <HealthStatus>Healthy</HealthStatus>
          <AvailabilityZone>us-east-1e</AvailabilityZone>
          <InstanceId>i-5a277829</InstanceId>
          <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
          <LifecycleState>InService</LifecycleState>
        </member>
      </Instances>
      <DesiredCapacity>1</DesiredCapacity>
      <AvailabilityZones>
        <member>us-east-1e</member>
      </AvailabilityZones>
      <LoadBalancerNames/>
      <MinSize>1</MinSize>
      <VPCZoneIdentifier/>
      <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
      <DefaultCooldown>300</DefaultCooldown>
      <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:bf6a47f1-1eda-4108-bf48-8d8da0c2b72e:auto
ScalingGroupName/my-test-asg</
AutoScalingGroupARN>
      <TerminationPolicies>
        <member>Default</member>
      </TerminationPolicies>
      <MaxSize>10</MaxSize>
    </member>
  </AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>7e20e72f-851c-11e2-b688-ff8f1EXAMPLE</RequestId>
```

```
</ResponseMetadata>  
</DescribeAutoScalingGroupsResponse>
```

Take a look at the `member` fields. Your Auto Scaling group now has one EC2 instance.

You've successfully created Auto Scaling policies and CloudWatch alarms to scale based on the number of messages in the Amazon SQS queue waiting to be processed. You have associated your scaling policies with the CloudWatch alarms. And you have tested to verify that your Auto Scaling group scales when the specified conditions occur.

Controlling Access to Your Auto Scaling Resources

Auto Scaling integrates with AWS Identity and Access Management (IAM), a service that lets your organization do the following:

- Create users and groups under your organization's AWS account
- Easily share your AWS account resources between the users in the account
- Assign unique security credentials to each user
- Granularly control users access to services and resources
- Get a single AWS bill for all users under the AWS account

For example, you can use IAM with Auto Scaling to control which users in your AWS account can create launch configurations, Auto Scaling groups, or Auto Scaling policies.

For general information about IAM, see:

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

Use IAM with Auto Scaling

Auto Scaling does not offer its own resource-based permissions system. However, Auto Scaling integrates with IAM so that you can specify which actions a user in your AWS account can perform with Auto Scaling resources in general. However, you can't specify a particular Auto Scaling resource in the policy (such as, a specific Auto Scaling group). For example, you could create an IAM policy that gives the Managers group permission to use only `DescribeAutoScalingGroups`, `DescribeLaunchConfigurations`, `DescribeScalingActivities`, and `DescribePolicies`. They could then use those actions with any Auto Scaling groups and launch configurations that belong to your AWS account.

Important

Using Auto Scaling with IAM doesn't change how you use Auto Scaling. There are no changes to Auto Scaling actions, and no new Auto Scaling actions related to users and access control.

For examples of IAM policies that cover Auto Scaling actions and resources, see [Example IAM Policies for Auto Scaling](#) (p. 106).

Topics

- [Auto Scaling ARNs](#) (p. 106)

- [Auto Scaling Actions](#) (p. 106)
- [Auto Scaling Keys](#) (p. 106)
- [Example IAM Policies for Auto Scaling](#) (p. 106)

Auto Scaling ARNs

Auto Scaling has no ARNs for you to use because you can't specify a particular Auto Scaling resource in an Auto Scaling policy. When writing an IAM policy to control access to Auto Scaling actions, you use `*` as the resource. For more information, see [ARNs](#).

Auto Scaling Actions

In an Auto Scaling policy, you can specify any and all actions that Auto Scaling offers. The action name must be prefixed with the lowercase string `autoscaling:`. For example: `autoscaling:CreateAutoScalingGroup`, `autoscaling:CreateLaunchConfiguration`, `autoscaling:*` (for all Auto Scaling actions). For a list of the actions, refer to the API action names in the [Auto Scaling API Reference](#).

Auto Scaling Keys

Auto Scaling implements the following policy keys, but no others. For more information about policy keys, see [Available Keys](#) in the Conditions section of Element Descriptions topic.

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:MultiFactorAuthAge`—To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used, this key is not present.
- `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:user-id`—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

This section shows several simple IAM policies for controlling user access to Auto Scaling.

Note

In the future, Auto Scaling might add new actions that should logically be included in one of the following IAM policies, based on the policy's stated goals.

1: Allow a group to create and manage Auto Scaling launch configurations.

In this example, create an IAM policy that gives access to all Auto Scaling actions that include the literal string `LaunchConfiguration` in the name. The resource is stated as `"*"`, because you can't specify a particular Auto Scaling resource in an Auto Scaling policy.

Note

The policy uses wildcards to specify all actions for launch configurations. You could instead list each action explicitly. If you use the wildcards instead of explicitly listing each action, be aware that if new Auto Scaling actions whose names include the string `LaunchConfiguration` become available, the policy would automatically give the group access to those new actions.

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

2: Allow system administrators to create and manage Auto Scaling groups and policies.

In this example, create a group for system administrators, and assign an IAM policy that gives access to any of the Auto Scaling actions that include the literal string `Scaling` in the name.

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

3: Allow developers to change the capacity of an Auto Scaling group.

In this example, create a group for developers and assign an IAM policy that gives access to the `SetDesiredCapacity` action.

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

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

If you plan to use the IAM CLI, be sure to install the IAM CLI. For more information, see [AWS Identity and Access Management Command Line Interface Reference](#).

Steps for Launching Instances with an IAM role

After you have created the IAM role, the IAM instance profile, and have added the role to the instance profile, you are ready to launch Auto Scaling instances with the IAM role, using the following steps:

- Create a launch configuration by specifying the IAM instance profile name or the full ARN of the IAM instance profile.
- Create an Auto Scaling group with the launch configuration that you just created.
- Verify that the EC2 instance was launched with the IAM role.

Launching Instances with the CLI

Use the following Auto Scaling commands to launch instances.

Commands	Description
<code>as-create-launch-config</code>	Creates a new launch configuration with specified attributes.
<code>as-create-auto-scaling-group</code>	Creates a new Auto Scaling group with the specified name and other attributes.

Commands	Description
<code>as-describe-auto-scaling-groups</code>	Describes the Auto Scaling groups, if the groups exist.

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#). Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

For this procedure, specify the following values for the `as-create-launch-config` command:

- Launch configuration name = `lc-with-instance-profile`
- Image ID = `ami-baba68d3`
If you don't have an AMI, and you want to find a suitable one, follow the instructions in [Finding a Suitable AMI](#).
- Instance type = `m1.small`
- Instance profile name = `mytest-instance-profile`.

Your command should look similar to the following example:

```
as-create-launch-config lc-with-instance-profile --image-id ami-baba68d3 --in-  
stance-type m1.small --iam-instance-profile mytest-instance-profile
```

You should get a confirmation like the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Create your Auto Scaling group by using `as-create-auto-scaling-group` and then specifying the launch configuration you just created. For more information about the syntax of the `as-create-auto-scaling-group` command, see [Create an Auto Scaling Group \(p. 36\)](#).

Specify these values for the command:

- Auto Scaling group name = `asg-using-instance-profile`
- Launch configuration name = `lc-with-instance-profile`
- Availability Zone = `us-east-1e`
- Max size = 1
- Min size = 1

Your command should look similar to the following example:

```
as-create-auto-scaling-group asg-using-instance-profile --launch-configuration  
lc-with-instance-profile --availability-zones "us-east-1e" --max-size 1 --min-  
size 1
```

You should get confirmation similar to the following example:

```
OK-Created AutoScalingGroup
```

Verify That the EC2 Instance Launches with the IAM Role

To confirm that Auto Scaling launches your EC2 instances using the IAM role you specify, use `as-describe-auto-scaling-groups`. The command shows details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see [Verify Your Auto Scaling Group Creation \(p. 37\)](#).

Your command should look like the following example:

```
as-describe-auto-scaling-groups asg-using-instance-profile --headers
```

Note

Specify the `--headers` general option to show column headers that organize the describe command's information.

The information you get should be similar to the following example.

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY
AUTO-SCALING-GROUP	asg-using-instance-profile	lc-with-instance-profile	us-east-1e		1	1

INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG
INSTANCE	i-5d97a03b	us-east-1e		InService	Healthy lc-with-instance-profile

You can see that Auto Scaling launched an instance using the `lc-with-instance-profile` launch configuration; and it is running (`InService`) and is healthy.

Clean Up

After you're finished using your instances and your Auto Scaling group, it is a good practice to clean up. Run the `as-delete-auto-scaling-group` command with the optional `--force-delete` parameter. *Force delete* specifies that EC2 instances that are part of the Auto Scaling group are deleted with the Auto Scaling group, even if the instances are still running. If you don't specify the `--force-delete` parameter, then you cannot delete your Auto Scaling group until you have terminated all instances in that Auto Scaling group.

Run the command with the following values:

- Auto Scaling group name = `asg-with-instance-profile`
- Force delete (optional parameter) = `--force-delete`

Your command should look like the following example:

```
as-delete-auto-scaling-group asg-with-instance-profile --force-delete
```

Confirm that you want to delete the Auto Scaling group. After you confirm that you want to delete the Auto Scaling group, Auto Scaling deletes the group, as the following example shows:

```
Are you sure you want to delete this AutoScalingGroup? [Ny]
OK-Deleted AutoScalingGroup
```

Creating Launch Configurations

At the core of your Auto Scaling group is a launch configuration. A launch configuration is a template that applies to the EC2 instances launched into the Auto Scaling group.

If you've launched an individual EC2 instance before, you've already walked through the process of defining compute characteristics such as the instance type, security groups, and configuration scripts. A launch configuration allows you to define these same characteristics, which are then applied to any instances launched into the Auto Scaling group.

Any Auto Scaling group that you create requires a launch configuration. You can change which launch configuration a group uses at any time—a useful feature when you want to update the instances in an existing group. However, you can't modify a launch configuration after you've created it.

You can create a launch configuration manually, using one of the following options:

- AWS Management Console
- AWS CLI (`aws autoscaling create-launch-configuration`)
- Auto Scaling CLI
- `CreateLaunchConfig` API

The [Getting Started with Auto Scaling \(p. 28\)](#) section describes how to use both the AWS Management Console and the command line to create a launch configuration.

You also have the option to [create a launch configuration based on an existing EC2 instance \(p. 111\)](#). This option is helpful when you already have existing EC2 instances running in AWS.

Note

In addition, you can [create an Auto Scaling group directly from an EC2 instance \(p. 122\)](#). When you use this feature, Auto Scaling automatically creates a launch configuration for you.

When your launch configuration is ready, you can move straight to [creating your Auto Scaling group \(p. 122\)](#). However, you might also want to look at how to [control when your instances launch and terminate \(p. 137\)](#), [how to tag instances \(p. 158\)](#) so they're easier to identify, and [how to incorporate Spot Instances \(p. 162\)](#) to make your Auto Scaling group even more cost effective.

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 an existing 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.

To use this section you should be familiar with EC2 instances, with the EC2 instance attributes, and with the process for launching EC2 instances. For more information, see [Launching an 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 name of the launch configuration, Amazon Machine Image (AMI) ID of the image you want to use, and the type of instance you want to launch. You can optionally list additional resources such as storage

devices, identity and access management options, monitoring, and so on. For information on the attributes you can use to create a launch configuration, see [CreateLaunchConfiguration](#).

When you create a launch configuration from an existing instance, you specify the instance ID and the name of the launch configuration. Auto Scaling derives all the other attributes applicable to the launch configuration from the specified instance, with the exception of the `BlockDeviceMapping`. By default, Auto Scaling creates the new launch configuration using the block device mapping that comes with the AMI that was used to create the instance and ignores any additional block device mappings that was later added to the instance. For more information on block device mapping, see [Block Device Mapping](#) in the *Amazon EC2 User Guide for Linux Instances*.

You can override any instance attribute, including the block device mapping on the instance's AMI, by specifying your own as a part of the same request.

To get started creating a launch configuration using an EC2 instance, you need to identify the instance you want to use. You can identify the EC2 instance by looking at the descriptions of all the EC2 instances associated with your AWS account. Use the Amazon EC2 console in the AWS Management Console, the `ec2-describe-instances` Amazon EC2 command, or the [DescribeInstances](#) Amazon EC2 action to get the descriptions of all the instances associated with your AWS account. After you identify the instance, make a note of the instance ID and make sure to verify the following :

- The identified instance is in `running` state.
- The identified instance is in the same region as the one in which you want to create your new launch configuration.

Note

If the identified 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 may not be identical to the identified instance.

For this procedure, use instance `i-a8e09d9c` with the following attributes to create a new launch configuration:

- Instance ID = `i-a8e09d9c`
- Image ID = `ami-b8a63b88`
- Instance type = `t1.micro`
- Monitoring = `basic`
- Kernel ID = `aki-6065f250`
- Security group = `default`
- EBS-optimized = `false`
- Root device type = `ebs`
- Root device = `/dev/sda1`
- Block devices = `/dev/sda1` and `/dev/sdf`

Note

The block device `/dev/sda1` is the root device that comes with the AMI and the block device `/dev/sdf` is an Amazon EBS volume.

You can create a launch configuration from an EC2 instance using either the Auto Scaling command line interface (CLI) or the Query API. Currently, Auto Scaling in the Amazon EC2 console does not support this feature.

The following sections walk you through several ways to create a new launch configuration from an EC2 instance. Choose the one that suits your requirements.

Topics

- [Create a Launch Configuration Using an Instance ID \(p. 113\)](#)
- [Create a Launch Configuration Using Your Own Block Device Mapping \(p. 115\)](#)
- [Create a Launch Configuration by Overriding an Instance Attribute \(p. 119\)](#)

Create a Launch Configuration Using an Instance ID

This section walks you through several ways to create a new launch configuration `my-test-lc-from-instance` from an instance `i-a8e09d9c`. After you complete this procedure, your new launch configuration have all the attributes from the instance except `/dev/sdf`, the block device mapping of the Amazon EBS volume.

Using the Command Line Interface

This section walks you through the process of creating a new launch configuration from an instance using the Auto Scaling command line interface.

To create a launch configuration using an instance ID

1. Enter the `as-create-launch-config` command with the following parameters:

- Launch configuration name = `my-test-lc-from-instance`
- Instance ID = `i-a8e09d9c`

Create a launch configuration using the instance ID.

```
as-create-launch-config my-test-lc-from-instance --instance-id i-a8e09d9c
```

When the launch configuration is created, Auto Scaling returns a success message.

```
OK-Created launch config
```

2. Enter the `as-describe-launch-configs` command with the following parameter:

- Launch configuration name = `my-test-lc-from-instance`

Describe the launch configuration:

```
as-describe-launch-configs my-test-lc-from-instance --show-long
```

The following description describes the launch configuration:

```
LAUNCH-CONFIG,my-test-lc-from-instance,ami-b8a63b88,t1.micro,(nil),aki-6065f250,(nil),{/dev/sda1=snap-3decf207:6:true:standard},sg-d6b3dae6,2013-11-22T05:01:59.291Z,false,arn:aws:autoscaling:us-east-1a:605053316265:launchConfiguration:39c95671-708e-4cd2-8642-7fa491e3f114:launchConfigurationName/my-test-lc-from-instance,
```

```
(nil),(nil),false,(nil)
```

You can see that the descriptions of the new launch configuration matches the description of the instance used to create this launch configuration, with the exception of the block device mapping. The block device mapping of the new launch configuration consists of just the root device: `/dev/sda1=snap-3decf207`. The block device mapping `/dev/sdf` is not associated with this new launch configuration.

Using the Query API

This section walks you through the process of creating a new launch configuration from an instance using the Auto Scaling Query API.

To create a launch configuration using an instance ID

1. Use the `CreateLaunchConfiguration` action with the following parameters:

- `LaunchConfigurationName` = `my-test-lc-from-instance`
- `InstanceId` = `i-a8e09d9c`

Create a launch configuration using the instance ID:

```
http://autoscaling.us-east-1a.amazonaws.com/?InstanceId=i-a8e09d9c
&LaunchConfigurationName=my-test-lc-from-instance
&Version=2011-01-01
&Action=CreateLaunchConfiguration
&AUTHPARAMS
```

The following response is an example of successful creation of a launch configuration using instance ID:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>3bc50863-5334-11e3-a6c2-f3fEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

2. Use the `DescribeLaunchConfigurations` action with the following parameter:

- `LaunchConfigurationName` = `my-test-lc-from-instance`

Describe the launch configuration:

```
http://autoscaling.us-east-1a.amazonaws.com/?LaunchConfigurationName=my-
test-lc-from-instance
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeLaunchConfigurations
&AUTHPARAMS
```

The following response describes the launch configuration created using the instance ID:

```
<DescribeLaunchConfigurationsResponse xmlns="http://autoscaling
c/2011-01-01/">
  <DescribeLaunchConfigurationsResult>
    <LaunchConfigurations>
      <member>
        <SecurityGroups>
          <member>sg-d6b3dae6</member>
        </SecurityGroups>
        <CreatedTime>2013-11-22T05:01:59.291Z</CreatedTime>
        <KernelId>aki-6065f250</KernelId>
        <LaunchConfigurationName>my-test-lc-from-instance</LaunchConfigura
tionName>
        <UserData/>
        <LaunchConfigurationARN>arn:aws:autoscaling:us-east-
1a:605053316265:launchConfiguration:39c95671-708e-4cd2-8642-
7fa491e3f114:launchConfiest-lc-from-instance</LaunchConfigurationARN>
        <InstanceType>t1.micro</InstanceType>
        <BlockDeviceMappings>
          <member>
            <DeviceName>/dev/sda1</DeviceName>
            <Ebs>
              <SnapshotId>snap-3decf207</SnapshotId>
              <DeleteOnTermination>true</DeleteOnTermination>
              <VolumeSize>6</VolumeSize>
              <VolumeType>standard</VolumeType>
            </Ebs>
          </member>
        </BlockDeviceMappings>
        <ImageId>ami-b8a63b88</ImageId>
        <KeyName/>
        <RamdiskId/>
        <InstanceMonitoring>
          <Enabled>>false</Enabled>
        </InstanceMonitoring>
        <EbsOptimized>>false</EbsOptimized>
      </member>
    </LaunchConfigurations>
  </DescribeLaunchConfigurationsResult>
  <ResponseMetadata>
    <RequestId>9c6ab3cd-5334-11e3-88ee-51c99EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeLaunchConfigurationsResponse>
```

You can see that the descriptions of the new launch configuration matches the descriptions of the instance used to create this launch configuration, with the exception of the block device mapping. The block device mapping of the new launch configuration consists of just the root device: `/dev/sda1`. The Amazon EBS volume `/dev/sdf` is not associated with this new launch configuration.

Create a Launch Configuration Using Your Own Block Device Mapping

In the previous section, you created a new launch configuration from an EC2 instance by specifying the instance ID.

This section walks you through the process of creating a new launch configuration, `my-test-lc-from-instance-bdm`, from an instance, `i-a8e09d9c` by specifying your own block device mappings for the root device `/dev/sda1` and the Amazon EBS volume `/dev/sdf`.

Using the Command Line Interface

This section uses the command line interface (CLI) to walk you through the process of creating a new launch configuration using a block device mapping that you choose.

To create a launch configuration using your own block device mapping

1. Enter the `as-create-launch-config` command with the following parameters:
 - Launch configuration name = `my-test-lc-from-instance-bdm`
 - Instance ID = `i-a8e09d9c`
 - Block device mapping = `dev/sda1=snap3decf207,/dev/sdf=snap-eed6ac86:10`

Create launch configuration using your own block device mapping:

```
as-create-launch-config my-test-lc-from-instance-bdm --instance-id i-a8e09d9c
--block-device-mapping "/dev/sda1=snap3decf207,/dev/sdf=snap-eed6ac86:10"
```

When the launch configuration is created, Auto Scaling returns a success message:

```
OK-Created launch config
```

2. Enter the `as-describe-launch-configs` command with the following parameter:
 - Launch configuration name = `my-test-lc-from-instance-bdm`

Describe the launch configuration:

```
as-describe-launch-configs my-test-lc-from-instance-bdm --show-long
```

The following description describes the launch configuration created using your own block device mapping:

```
LAUNCH-CONFIG,my-test-lc-from-instance-bdm,ami-b8a63b88,t1.micro,(nil),aki-
6065f250,(nil),"{/dev/sda1=snap3decf207,/dev/sdf=snap-eed6ac86:10}",sg-
d6b3dae6,2013
-12-04T12:37:43.366Z,false,arn:aws:autoscaling:us-east-
1a:605053316265:launchConfiguration:849a3adf-d5f9-429e-8a33-
fa9669fcdb03:launchConfigurationName/my-test-l
c-from-instance-bdm,(nil),(nil),false,(nil)
```

You can see that the block device mapping of the new launch configuration shows both the root device `/dev/sda1` and the Amazon EBS volume `/dev/sdf`.

Using the Query API

This section uses the Query API to walk you through the process of creating a new launch configuration using a block device mapping that you choose.

To create a launch configuration using your own block device mapping

1. Use the `CreateLaunchConfiguration` action with the following parameters:

- `LaunchConfigurationName` = `my-test-lc-from-instance-bdm`
- `InstanceId` = `i-a8e09d9c`
- `BlockDeviceMapping.1.DeviceName` = `/dev/sda1`
`BlockDeviceMapping.1.Ebs.SnapshotId` = `snap3decf207`
`BlockDeviceMapping.2.DeviceName` = `/dev/sdf`
`BlockDeviceMapping.2.Ebs.SnapshotId` = `snap-eed6ac86`
`BlockDeviceMapping.2.Ebs.VolumeSize` = `10`

Create a launch configuration using your own block device mapping:

```
http://autoscaling.us-east-1a.amazonaws.com/?InstanceId=i-a8e09d9c
&BlockDeviceMappings.member.1.DeviceName=%2Fdev%2Fsda1
&BlockDeviceMappings.member.1.Ebs.SnapshotId=snap3decf207
&BlockDeviceMappings.member.2.Ebs.VolumeSize=10
&BlockDeviceMappings.member.2.DeviceName=%2Fdev%2Fsdf
&BlockDeviceMappings.member.2.Ebs.SnapshotId=snap-eed6ac86
&LaunchConfigurationName=my-test-lc-from-instance-bdm
&Version=2011-01-01
&Action=CreateLaunchConfiguration
&AUTHPARAMS
```

The following response is an example of successful creation of a launch configuration using your own block device mapping:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>3bc50863-5334-11e3-a6c2-f3fEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

2. Use the `DescribeLaunchConfigurations` action with the following parameter:

- `LaunchConfigurationName` = `my-test-lc-from-instance-bdm`

Describe the launch configuration:

```
http://autoscaling.us-east-1a.amazonaws.com/?LaunchConfigurationName=my-
test-lc-from-instance-bdm
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeLaunchConfigurations
&AUTHPARAMS
```

The following response describes the launch configuration created using your own block device mapping:

```
<DescribeLaunchConfigurationsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeLaunchConfigurationsResult>
    <LaunchConfigurations>
      <member>
        <SecurityGroups>
          <member>sg-d6b3dae6</member>
        </SecurityGroups>
        <CreatedTime>2013-12-04T12:37:43.366Z</CreatedTime>
        <KernelId>aki-6065f250</KernelId>
        <LaunchConfigurationName>my-test-lc-from-instance-bdm</LaunchConfig
urationName>
        <UserData/>
        <InstanceType>t1.micro</InstanceType>
        <LaunchConfigurationARN>arn:aws:autoscaling:us-east-
1a:605053316265:launchConfiguration:849a3adf-d5f9-429e-8a33-
fa9669fcdb03:launchConfigurationName/my-t
est-lc-from-instance-bdm</LaunchConfigurationARN>
        <BlockDeviceMappings>
          <member>
            <DeviceName>/dev/sda1</DeviceName>
            <Ebs>
              <SnapshotId>snap3decf207</SnapshotId>
            </Ebs>
          </member>
          <member>
            <DeviceName>/dev/sdf</DeviceName>
            <Ebs>
              <SnapshotId>snap-eed6ac86</SnapshotId>
              <VolumeSize>10</VolumeSize>
            </Ebs>
          </member>
        </BlockDeviceMappings>
        <KeyName/>
        <ImageId>ami-b8a63b88</ImageId>
        <RamdiskId/>
        <InstanceMonitoring>
          <Enabled>>false</Enabled>
        </InstanceMonitoring>
        <EbsOptimized>>false</EbsOptimized>
      </member>
    </LaunchConfigurations>
  </DescribeLaunchConfigurationsResult>
  <ResponseMetadata>
    <RequestId>ddfc67b7-5ce1-11e3-bdb1-d96ecEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeLaunchConfigurationsResponse>
```

You can see that the block device mapping of the new launch configuration shows both the root device `/dev/sda1` and the Amazon EBS volume `/dev/sdf`.

Create a Launch Configuration by Overriding an Instance Attribute

By default, Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration. Depending on your requirements, you might not want to use some of the instance attributes. Auto Scaling provides you with an option to override the instance attributes that do not work for you.

This section walks you through the process of creating a new launch configuration `my-test-lc-from-instance-changetype` from an instance `i-a8e09d9c` by specifying `m1.small` for the instance type. Note that instance type for `i-a8e09d9c` is a `t1.micro`. This procedure overrides the instance type.

Using the Command Line Interface

This section uses the command line interface (CLI) to walk you through the process of using an EC2 instance to create a new launch configuration but changing the instance type.

To create a launch configuration using a different instance type

1. Enter the `as-create-launch-config` command with the following parameters:

- Launch configuration name = `my-test-lc-from-instance-changetype`
- Instance ID = `i-a8e09d9c`
- Instance type = `m1.small`

Create a launch configuration using your own instance type.

```
as-create-launch-config my-test-lc-from-instance-changetype --instance-id  
i-a8e09d9c --instance-type m1.small
```

When the launch configuration is created, Auto Scaling returns a success message.

```
OK-Created launch config
```

2. Enter the `as-describe-launch-configs` command with the following parameter:

- Launch configuration name = `my-test-lc-from-instance-changetype`

Describe the launch configuration:

```
as-describe-launch-configs my-test-lc-from-instance-changetype --headers  
--show-long
```

The following description describes the launch configuration:

```
LAUNCH-CONFIG,NAME,IMAGE-ID,TYPE,KEY,KERNEL-ID,RAMDISK-ID,DEVICE-  
MAP,SG,CREATED,  
MONITORING,LAUNCH-CONFIG-ARN,SPOT-PRICE,IAM-INSTANCE-PROFILE,EBS-OPTIMIZED,AS  
SOC  
IATE-PUBLIC-IP-ADDRESS  
LAUNCH-CONFIG,my-test-lc-from-instance-changetype,ami-  
b8a63b88,m1.small,(nil),aki-6065f25  
0,(nil),{/dev/sda1=snap-3decf207:6:true:standard},sg-d6b3dae6,2013-11-
```

```
22T05:22:1
3.556Z,false,arn:aws:autoscaling:us-east-1a:605053316265:launchConfigura
tion:8d91
a48c-7f35-4572-a913-5b0caa86b975:launchConfigurationName/my-test-lc-from-
instance-changetype,(nil),(nil),false,(nil)
```

Check the configurations of the instance `i-a8e09d9c`. The instance configurations should match the descriptions of the new launch configuration with the exception of the instance type and the block device mapping. In the new launch configuration, the instance type is changed to `m1.small`.

Using the Query API

This section uses the Query API to walk you through the process of using an EC2 instance to create a new launch configuration but changing the instance type.

To create a launch configuration using a different instance type

1. Use the `CreateLaunchConfiguration` action with the following parameters:

- `LaunchConfigurationName` = `my-test-lc-from-instance-changetype`
- `InstanceId` = `i-a8e09d9c`
- `InstanceType` = `m1.small`

Create a launch configuration using your own instance type:

```
http://autoscaling.us-east-1a.amazonaws.com/?InstanceId=i-a8e09d9c
&InstanceType=m1.small
&LaunchConfigurationName=my-test-lc-from-instance-changetype
&Version=2011-01-01
&Action=CreateLaunchConfiguration
&AUTHPARAMS
```

The following response is an example of successful creation of a launch configuration using your own instance type:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>3bc50863-5334-11e3-a6c2-f3fEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

2. Use the `DescribeLaunchConfigurations` action with the following parameter:

- `LaunchConfigurationName` = `my-test-lc-from-instance-changetype`

Describe the launch configuration

```
http://autoscaling.us-east-1a.amazonaws.com/?LaunchConfigurationName=my-
test-lc-from-instance-changetype
&MaxRecords=20
&Version=2011-01-01
```

```
&Action=DescribeLaunchConfigurations
&AUTHPARAMS
```

The following response describes the launch configuration created using your own instance type:

```
<DescribeLaunchConfigurationsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeLaunchConfigurationsResult>
    <LaunchConfigurations>
      <member>
        <SecurityGroups>
          <member>sg-d6b3dae6</member>
        </SecurityGroups>
        <CreatedTime>2013-11-22T05:22:13.556Z</CreatedTime>
        <KernelId>aki-6065f250</KernelId>
        <LaunchConfigurationName>my-test-lc-from-instance-changetype</Launch
ConfigurationName>
        <UserData/>
        <InstanceType>m1.small</InstanceType>
        <LaunchConfigurationARN>arn:aws:autoscaling:us-east-
1a:605053316265:launc
hConfiguration:8d91a48c-7f35-4572-a913-5b0caa86b975:launchConfiguration
Name/my-test-lc-from-instance-changetype</LaunchConfigurationARN>
        <BlockDeviceMappings>
          <member>
            <DeviceName>/dev/sda1</DeviceName>
            <Ebs>
              <SnapshotId>snap-3decf207</SnapshotId>
              <DeleteOnTermination>true</DeleteOnTermination>
              <VolumeSize>6</VolumeSize>
              <VolumeType>standard</VolumeType>
            </Ebs>
          </member>
        </BlockDeviceMappings>
        <KeyName/>
        <ImageId>ami-b8a63b88</ImageId>
        <RamdiskId/>
        <InstanceMonitoring>
          <Enabled>>false</Enabled>
        </InstanceMonitoring>
        <EbsOptimized>>false</EbsOptimized>
      </member>
    </LaunchConfigurations>
  </DescribeLaunchConfigurationsResult>
  <ResponseMetadata>
    <RequestId>5e4339a4-5336-11e3-8066-d99e2EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeLaunchConfigurationsResponse>
```

Check the configurations of the instance `i-a8e09d9c`. The instance configurations should match the descriptions of the new launch configuration with the exception of the instance type and the block device mapping. In the new launch configuration, the instance type is changed to `m1.small`.

Creating Your 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.

Note

If you're new to creating Auto Scaling groups, we recommend you read the [Getting Started with Auto Scaling \(p. 28\)](#) section, as well as the [Planning your Auto Scaling Group \(p. 44\)](#) section, to learn more about how Auto Scaling works and what you should consider before adding Auto Scaling to your network architecture.

At a minimum, an Auto Scaling group must contain:

- 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

Of course, an Auto Scaling group that only has these parameters defined does not provide much value. Some additional configuration options that you'll want to define include:

- 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 the section, [Creating Launch Configurations \(p. 111\)](#), you need to define what instance type the Auto Scaling group uses and how each instance is 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.

These are just a few of the options available to you. Check out the [CreateAutoScalingGroup](#) action for a complete list.

Note

You also have the option of creating an Auto Scaling directly from an existing EC2 instance. This is helpful when you have an existing application and want to test out Auto Scaling. You can always [detach the instances \(p. 208\)](#) from the Auto Scaling group if you need to.

The [Getting Started with Auto Scaling \(p. 28\)](#) section describes how to create a basic Auto Scaling group. If you're creating an Auto Scaling group inside a VPC, review the section, [Auto Scaling in Amazon Virtual Private Cloud \(p. 128\)](#). In addition, you might want to look at the section [Configuring Your Auto Scaling Groups \(p. 184\)](#) for actions that you can take when the group is in production. Also, the section, [Monitoring Your Auto Scaling Instances \(p. 241\)](#) provides more details on how to track the performances of instances in the Auto Scaling group.

Create an Auto Scaling Group Using an EC2 Instance ID

Auto Scaling provides you with an option of creating an Auto Scaling group by specifying an EC2 instance ID, instead of a launch configuration, and by specifying attributes such as the minimum, the maximum, and the 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. The launch configuration takes the name of the Auto Scaling group and derives its attributes from the specified instance, with the exception of `BlockDeviceMapping`.

For information about the attributes used to create a launch configuration, see [CreateLaunchConfiguration](#).

If you want to create an Auto Scaling group with block device mappings, first create a launch configuration by specifying the block device mappings and then create the Auto Scaling group using the launch configuration you just created.

This section includes the procedures to create a new Auto Scaling group using an EC2 instance. Before you begin, be sure you've completed the following steps:

- Identified an EC2 instance to use to create your new Auto Scaling group. You can identify the instance by looking at the descriptions of all the EC2 instances associated with your AWS account. Use the AWS Management Console, the `ec2-describe-instances` Amazon EC2 command, or the `DescribeInstances` Amazon EC2 action to get the descriptions of all the instances associated with your AWS account. After you've identified the instance, make a note of the instance ID, and make sure to verify the following:
 - The identified instance is in `running` state.
 - The identified instance is in the same Availability Zone as the one in which you want to create your new Auto Scaling group.
 - The identified instance and the new Auto Scaling group belong to the same AWS account.
 - The identified instance is not a member of another Auto Scaling group.

For this procedure, you use instance `i-7f12e649` with the following attributes to create a new Auto Scaling group:

- Instance ID = `i-7f12e649`
- Image ID = `ami-be1c848e`
- Instance type = `t1.micro`
- Availability Zone = `us-east-1a`
- Monitoring = `basic`
- Security group = `default`
- EBS-optimized = `false`
- Kernel ID = `aki-fc8f11cc`
- Root device type = `ebs`
- Root device = `/dev/sda1`

Note

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

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.

You can create an Auto Scaling group from an EC2 instance using either the Auto Scaling command line interface (CLI) or the Query API. Currently, the Auto Scaling console does not support this feature.

Topics

- [Create an Auto Scaling Group with an Instance ID Using the Command Line Interface \(p. 124\)](#)
- [Create an Auto Scaling Group with an Instance ID Using the Query API \(p. 125\)](#)

Create an Auto Scaling Group with an Instance ID Using the Command Line Interface

In this section, you create a new Auto Scaling group `my-asg-from-instance` from EC2 instance `i-7f12e649` using the Auto Scaling CLI.

To create an Auto Scaling group using an instance ID

1. Enter the `as-create-auto-scaling-group` command with the following parameters:

- Auto Scaling group name = `my-asg-from-instance`
- Instance ID = `i-7f12e649`
- Minimum size = `1`
- Maximum size = `2`
- Desired capacity = `2`

Create an Auto Scaling group using the instance ID.

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

When the Auto Scaling group is created, Auto Scaling returns a success message.

```
OK-Created AutoScalingGroup
```

2. Enter the `as-describe-auto-scaling-groups` command with the following parameter:

- Auto Scaling group name = `my-asg-from-instance`

Describe the Auto Scaling group:

```
as-describe-auto-scaling-groups my-asg-from-instance --headers
```

The following description describes the Auto Scaling group:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-asg-from-instance	my-asg-from-instance	us-east-1a	1	2	2	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-08d6a93e	us-east-1a	InService	Healthy	my-asg-from-instance		
INSTANCE	i-0bd6a93d	us-east-1a	InService	Healthy	my-asg-from-instance		

You can see that Auto Scaling has created a new launch configuration with the same name as the Auto Scaling group `my-asg-from-instance` and has launched new instances in `us-east-1a`, the same Availability Zone as the instance `i-7f12e649`.

3. Enter the `as-describe-launch-configs` command with the following parameter to see the description of the newly created launch configuration.

- Launch configuration name = *my-asg-from-instance*

Describe launch configuration.

```
as-describe-launch-configs my-asg-from-instance --show-long --headers
```

The following description describes the launch configuration.

```
LAUNCH-CONFIG,NAME,IMAGE-ID,TYPE,KEY,KERNEL-ID,RAMDISK-ID,DEVICE-  
MAP,SG,CREATED,MONITRING,LAUNCH-CONFIG-ARN,SPOT-PRICE,IAM-INSTANCE-PROFILE,EBS-OPTIMIZED,ASSOCIATE-PUBLIC-IP-  
ADDRESS  
LAUNCH-CONFIG,my-asg-from-instance,ami-belc848e,t1.micro,my-default-key,aki-  
fc8f11cc,(nil),(nil),sg-74033044,2013-12-12T18:12:38.550Z,false,arn:aws:autoscaling:us-east-1a:605053316265:launchConfiguration:a95ba162-a046-4376-8c04-d6707e62ab92:launchConfigurationName/my-asg-from-instance,(nil),(nil),false,(nil)
```

You can see the descriptions of the newly created launch configuration *my-asg-from-instance* match the descriptions of the instance *i-7f12e649*.

Create an Auto Scaling Group with an Instance ID Using the Query API

In this section, you create a new Auto Scaling group *my-asg-from-instance* from EC2 instance *i-7f12e649* using the Auto Scaling Query API.

To create a launch configuration using an instance ID

1. Use the [CreateAutoScalingGroup](#) action with the following parameters:

- AutoScalingGroupName = *my-asg-from-instance*
- Instance ID = *i-7f12e649*
- MinSize = *1*
- MaxSize = *2*
- DesiredCapacity = *2*

Create an Auto Scaling group using the instance ID:

```
http://autoscaling.us-east-1a.amazonaws.com/?AutoScalingGroupName=my-asg-  
from-instance  
&InstanceId=i-7f12e649  
&MinSize=1  
&MaxSize=2  
&DesiredCapacity=2  
&Version=2011-01-01  
&Action=CreateAutoScalingGroup  
&AUTHPARAMS
```

The following response is an example of successful creation of an Auto Scaling group using the instance ID:

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>1be2d1de-5d3b-11e3-8e1c-a924dfEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

2. Use the [DescribeAutoScalingGroups](#) action with the following parameter:

- AutoScalingGroupName = *my-asg-from-instance*

Describe the Auto Scaling group:

```
http://autoscaling.us-east-1a.amazonaws.com/?AutoScalingGroupNames.member.1=my-asg-from-instance
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The following response describes the Auto Scaling group created using the instance ID:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-asg-from-instance</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-12-12T18:12:38.563Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-asg-from-instance</LaunchConfiguration
Name>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-08d6a93e</InstanceId>
            <LaunchConfigurationName>my-asg-from-instance</LaunchConfigura
tionName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-0bd6a93d</InstanceId>
            <LaunchConfigurationName>my-asg-from-instance</LaunchConfigura
tionName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
</DescribeAutoScalingGroupsResponse>
```

```
</Instances>
<DesiredCapacity>2</DesiredCapacity>
<AvailabilityZones>
  <member>us-east-1a</member>
</AvailabilityZones>
<LoadBalancerNames/>
<MinSize>1</MinSize>
<VPCZoneIdentifier/>
<HealthCheckGracePeriod>0</HealthCheckGracePeriod>
<DefaultCooldown>300</DefaultCooldown>
<AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:605053316265:autoScalingGroup:9208116a-ac5f-40c8-b6fb-63078e60a7d7:autoScalingGroupName/my-asg-from-instance</AutoScalingGroupARN>
<TerminationPolicies>
  <member>Default</member>
</TerminationPolicies>
<MaxSize>2</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>85480448-635b-11e3-8868-27b198765a4b</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

You can see that Auto Scaling has created a new launch configuration with the same name as the Auto Scaling group *my-asg-from-instance* and has launched 2 new instances in us-east-1a, the same Availability Zone as the instance *i-7f12e649*.

3. Use the [DescribeLaunchConfigurations](#) action with the following parameter to see the description of the newly created launch configuration.
 - LaunchConfigurationName = *my-asg-from-instance*

Describe the launch configuration:

```
http://autoscaling.us-east-1a.amazonaws.com/?LaunchConfigurationNames.member.1=my-asg-from-instance
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeLaunchConfigurations
&AUTHPARAMS
```

The following response describes the launch configuration:

```
<DescribeLaunchConfigurationsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeLaunchConfigurationsResult>
    <LaunchConfigurations>
      <member>
        <SecurityGroups>
          <member>sg-ac33009c</member>
        </SecurityGroups>
        <CreatedTime>2013-12-12T07:02:14.873Z</CreatedTime>
        <KernelId>aki-fc8f11cc</KernelId>
      </member>
    </LaunchConfigurations>
  </DescribeLaunchConfigurationsResult>
</DescribeLaunchConfigurationsResponse>
```

```
<LaunchConfigurationName>my-asg-from-instance</LaunchConfiguration
Name>
  <UserData/>
  <InstanceType>t1.micro</InstanceType>
  <LaunchConfigurationARN>arn:aws:autoscaling:us-east-
1a:605053316265:launc
hConfiguration:f80882da-8411-40d6-8fa0-5ce7b1f7e81c:launchConfiguration
Name/my-asg-from-instance</LaunchConfigurationARN>
  <BlockDeviceMappings/>
  <KeyName>my-default-key</KeyName>
  <ImageId>ami-belc848e</ImageId>
  <RamdiskId/>
  <InstanceMonitoring>
    <Enabled>>false</Enabled>
  </InstanceMonitoring>
  <EbsOptimized>>false</EbsOptimized>
</member>
</LaunchConfigurations>
</DescribeLaunchConfigurationsResult>
<ResponseMetadata>
  <RequestId>b79bb607-6352-11e3-82cd-19b9EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeLaunchConfigurationsResponse>
```

You can see the descriptions of the newly created launch configuration *my-asg-from-instance* matches the descriptions of the instance *i-7f12e649*.

Auto Scaling in 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. Within this virtual private cloud, 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 \(p. 28\)](#) or see [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#).

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 and attach an Internet gateway 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.

Topics

- [Default VPC \(p. 129\)](#)
- [IP Addressing in a VPC \(p. 129\)](#)
- [Choosing Your Instance Placement Tenancy \(p. 129\)](#)
- [Related Topics \(p. 130\)](#)
- [Launch Auto Scaling Instances in a VPC \(p. 130\)](#)

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 \(p. 28\)](#).

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.

Choosing Your Instance Placement Tenancy

Instances launched by your Auto Scaling group in the VPC run on shared hardware, by default. This is shared-tenant instance placement. Auto Scaling provides you with the option to launch instances in a VPC that runs on hardware dedicated to a single customer. This is single-tenant instance placement. The instances that run on single-tenant hardware are called *Dedicated Instances*. 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, the instance tenancy attribute on the VPC is set to `default`. In this VPC, you can either launch instances that run on a shared-tenant hardware or instances that run on a single-tenant hardware. You can set the instance tenancy attribute to `dedicated` if you want the instances launched in the VPC to run on a single-tenant hardware.

When you create a launch configuration, depending on your requirement, you can either specify the tenancy attribute for your launch configuration or leave it set to null. If you do specify, you can set it to either `default` or `dedicated`.

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

LC Tenancy Attribute Value	Shared Tenancy VPC	Single Tenancy VPC
Unspecified	Shared tenancy instance	Dedicated instance
Default	Shared tenancy instance	Dedicated instance
Dedicated	Dedicated instance	Dedicated instance

Note

You cannot run Spot Instances or t1.micro instances on a dedicated hardware in a VPC.

Auto Scaling supports specifying instance tenancy attribute when you create launch configuration using the Auto Scaling command line interface (CLI), the AWS CLI, the Query API, or the AWS SDKs. Currently, you cannot specify instance tenancy attribute when you create launch configuration using the AWS Management Console, or when you use an AWS CloudFormation template.

There is a separate pricing model for Dedicated Instances. For more information, see the [Amazon EC2 Dedicated Instances product page](#).

Related Topics

- [Launch Auto Scaling Instances in a VPC \(p. 130\)](#)
- [What is Amazon VPC](#)
- [Your VPC and Subnets](#).

Launch Auto Scaling Instances in a VPC

Use the following procedures for launching your basic Auto Scaling infrastructure in a VPC. Before you begin, be sure you have signed up for an AWS account and have followed the instructions to install and use the interface of your choice to access Auto Scaling. For information about getting an AWS account or about the interfaces you can use, see [Setting Up Auto Scaling Interfaces \(p. 14\)](#).

Topics

- [Create a VPC \(p. 130\)](#)
- [Create a Launch Configuration \(p. 133\)](#)
- [Create an Auto Scaling Group \(p. 134\)](#)
- [Confirm that Instances Launched in Subnets \(p. 136\)](#)

Create a VPC

Before you can launch your Auto Scaling instances in a VPC, you must first create your VPC environment. You create a 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). Your VPC can span multiple Availability Zones and then you create one or more subnets in each Availability Zone. Subnets are defined by a segment of the IP address range of the VPC. After you create your VPC and subnets, you launch Auto Scaling instances within the subnets.

You can create your VPC environment using the AWS Management Console, the Amazon VPC command line interface (CLI), the AWS CLI, the Query API, or the SDKs.

Topics

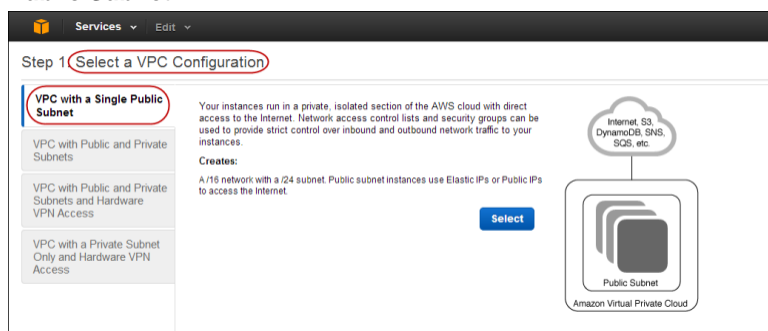
- [Launch in a VPC Using the AWS Management Console \(p. 131\)](#)
- [Launch in a VPC Using the Command Line Interface \(p. 132\)](#)

Launch in a VPC Using the AWS Management Console

In this section, use the VPC wizard in the Amazon VPC console to create a VPC and one public subnet.

To create a VPC with one public subnet

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. The Amazon VPC and Auto Scaling resources you create are tied to a region that you specify and are not replicated across regions. If necessary, change the region. From the navigation bar, select the region that meets your needs.
3. If the **VPC Dashboard** lists the resources you are currently using in the region, click **Start VPC Wizard**.
4. If this is the first time you are using Amazon VPC in the selected region, the **VPC Dashboard** page does not list any VPC resources. Click **Get Started With VPC**.
5. The VPC wizard opens. In the **Step 1: Select a VPC Configuration** page, select **VPC with a Single Public Subnet**.



6. Click **Select**.
7. The **Step 2: VPC with a Single Public Subnet** page shows the CIDR range that to use for your VPC and subnet (10.0.0.0/16 and 10.0.0.0/24, respectively) and other settings.

In the **VPC name** field, enter a name for your VPC.
8. You can optionally choose an Availability Zone for your public subnet in the **Public subnet** field and enter a name for your public subnet in the **Subnet name** field.
9. Leave the **Enable DNS hostnames** and set to default settings for this procedure.
10. If you want to launch Dedicated Instances, click the **Hardware tenancy** dialog box and select **Dedicated**.
11. Click **Create VPC** to create your VPC, subnet, Internet gateway, and route table.
12. A status window shows the work in progress. When the work completes, a status window confirms that your VPC has been successfully created. Click **OK** to close the status window and return to the VPC dashboard.
13. In the navigation pane, click **Your VPCs**, and then select the VPC that you just created.
14. The bottom pane displays the details of your VPC. Make a note of the VPC ID. You need it for the next step.

Launch in a VPC Using the Command Line Interface

In this procedure, you use Amazon EC2 CLI commands to create a VPC and a subnet. Before you begin, make sure you have installed the Amazon EC2 CLI tools. For instructions to download and install the Amazon EC2 tools, see [Setting Up the Amazon EC2 CLI and AMI Tools](#) in the *Amazon EC2 API Reference*.

To create a VPC

1. Use the `ec2-create-vpc` command as in the following example:

```
ec2-create-vpc "10.0.0.0/16"
```

The response includes your VPC ID.

```
VPC vpc-2e00c747 available 10.0.0.0/16 dopt-2200c74b default
```

2. To create a VPC with `dedicated` tenancy option, use the `ec2-create-vpc` command as in the following example.

```
ec2-create-vpc 10.0.0.0/16 --tenancy dedicated
```

The response includes your VPC ID.

```
VPC vpc-2e00c747 available 10.0.0.0/16 dopt-2200c74b dedicated
```

To create a subnet

1. Use the `ec2-create-subnet` command and specify the following parameters:

- `vpc = vpc-2e00c747`
- `cidr = "10.0.0.0/24"`
- `availability-zone = us-east-1a`

Your command should look similar to the following example:

```
ec2-create-subnet --vpc_id vpc-2e00c747 --cidr "10.0.0.0/24" --zone us-east-1a
```

2. The response includes your subnet ID, current state of the subnet (`available` or `pending`), the ID of the VPC the subnet is in, the number of IP addresses available in your subnet (`11`), and the Availability Zone the subnet is in.

```
SUBNET subnet-530fc83a available vpc-2e00c747 10.0.0.0/24 11 us-east-1a
```

3. Make a note of the subnet ID. You need it later when you create your Auto Scaling group.

Create a Launch Configuration

After you've created your VPC and subnets, you configure your Auto Scaling setup by first creating a launch configuration. The Auto Scaling launch configuration specifies the template that Auto Scaling uses to launch EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. Skip this step if you already have a launch configuration and want to use it to launch Auto Scaling instances in a VPC.

This section walks you through the steps for creating a new launch configuration using either the AWS Management Console or the Auto Scaling CLI.

Currently the AWS Management Console does not support choosing instance placement tenancy for your instances. If you are planning on specifying placement tenancy option for your instances launched using this launch configuration, follow the CLI instructions to create your launch configuration.

Topics

- [Creating a Launch Configuration Using the AWS Management Console \(p. 133\)](#)
- [Create a Launch Configuration Using the Command Line Interface \(p. 134\)](#)

Creating a Launch Configuration Using the AWS Management Console

To create a launch configuration

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the **Resources** page, in the left navigation pane, under **Auto Scaling**, click **Launch Configurations** to start the Auto Scaling wizard.
3. If the Auto Scaling wizard lists the launch configurations you have created in this region, click **Create Launch Configuration**.
4. If this is the first time you are using the Amazon EC2 console to access Auto Scaling in the selected region, your Auto Scaling wizard will not have any Launch Configurations. Click **Create Auto Scaling Group** and then click **Create Launch Configuration**.
5. In the **Create Launch Configuration** wizard, 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 or any other AMI that meets your requirements.
6. On the **2. Choose Instance Type** page, select the hardware configuration of your instance. This procedure uses the `t1.micro` instance that is selected by default. Click **Next: Configure details**.
7. On the **3. Configure Details** page, in the **Name** field, enter the name of your launch configuration (*my-vpc-lc*).
8. If you are not planning on assigning a public IP address to the instances launched in the VPC, leave the other fields blank for this procedure and click **Next: Add Storage**.

To assign a public IP address to your instances, complete the following steps:

- a. In the **Advanced Details** pane, leave the **Kernel ID**, **RAM Disk ID**, and **User data** fields blank for this tutorial.
 - b. In the **IP Address Type** field, choose the option that best suits your use case.
9. On the **4. Add Storage** page, add additional storage devices. This tutorial uses the storage devices that come with the selected AMI. Click **Next: Configure Security Group**.
 10. On **5. Configure Security Group** page, select **Select an existing new Security Group** and select the security group that is associated with your VPC. The VPC in the **VPC ID** column should match with the ID of your newly created VPC, and the **Description** column should indicate that the selected security group is a default VPC security group.

11. On the **Review** page, verify the details of your launch configuration. If you want to change any details, click **Edit** for the field you want to change.
12. After you are done reviewing your launch configuration, click **Create launch configuration**.
13. In the **Select an existing key pair or create a new key pair** field, select one of the listed options. Or, select **Proceed without a key pair**. This option is acceptable for this procedure.
14. Select the acknowledgement check box.
15. Click **Create Launch Configuration** to create your launch configuration.
16. 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 a Launch Configuration Using the Command Line Interface

In this step, you use the Auto Scaling CLI to create your launch configuration. Before you proceed, make sure you have installed the Auto Scaling CLI. If you have not installed the CLI, see [Install the Auto Scaling CLI \(p. 16\)](#) and install the CLI now.

To create a launch configuration

1. Use the `as-create-launch-config` command and specify the following parameters:

- Launch configuration name: `my-vpc-lc`
- Instance type: `m1.small`
- Image ID: `ami-b4b0cae6`

Note

The AMI ID is provided for illustration purposes only. AMI IDs change over time. You can obtain current, valid AMI IDs by calling the `ec2-describe-images` command. Make sure that the AMI or image ID that you specify is supported in the region of your VPC.

- [optional] Associate public IP address: `true`
- [optional] Placement tenancy: `dedicated`

Your command should look similar to the following example:

```
as-create-launch-config my-vpc-lc --image-id ami-b4b0cae6 --instance-type m1.small --associate-public-ip-address true --placement-tenancy dedicated
```

2. You get a confirmation that your launch configuration was created successfully.

```
OK-Created launch config
```

Create an Auto Scaling Group

An Auto Scaling group is a collection of EC2 instances. You can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions.

This section walks you through the steps for creating an Auto Scaling group using either the AWS Management Console or the Auto Scaling command line interface (CLI).

Topics

- [Create an Auto Scaling Group Using the AWS Management Console \(p. 135\)](#)
- [Create an Auto Scaling Group Using the Command Line Interface \(p. 135\)](#)

Create an Auto Scaling Group Using the AWS Management Console

To create an Auto Scaling group

1. On the **1. Configure Auto Scaling group** details page, enter the following details:
 - a. In the **Group name** field, enter a name for your Auto Scaling group (`my-vpc-asg`).
 - b. Leave the **Group size** field set to the default value of 1 instance for this procedure.
 - c. Click the **Network** field and select the VPC you created for this tutorial.
 - d. Click the **Subnet** field and select the subnet listed for the selected VPC.

The screenshot shows the 'Create Auto Scaling Group' wizard in the AWS Management Console. The first step, '1. Configure Auto Scaling group details', is active. The form contains the following fields and values:

- Launch Configuration:** my-vpc-lc
- Group name:** my-vpc-asg
- Group size:** Start with 1 instances
- Network:** vpc-ee64748c (10.0.0.0/16)
- Subnet:** subnet-1ab7ee5c (10.0.0.0/24) | ap-northeast-1c x

Buttons for 'Create new VPC' and 'Create new subnet' are visible next to the Network and Subnet fields respectively.

2. Click **Next: Configure scaling policies**.
3. On the **Configure scaling policies** page, select **Keep this group at its initial size** for this procedure.

If you want to configure scaling policies for your Auto Scaling group, see [Dynamic Scaling \(p. 57\)](#) and follow the instructions for creating scaling policies and CloudWatch alarms using the console.
4. Skip the next step for configuring notifications and click **Review** to verify your Auto Scaling group.
5. Review the details of your Auto Scaling group. You can click **Edit** on the right to edit the details.
6. Click **Create Auto Scaling Group**.
7. The **Auto Scaling Group creation status** page lets you know that your Auto Scaling group was successfully created. Click **Close**.

Create an Auto Scaling Group Using the Command Line Interface

To create an Auto Scaling group

1. Use `as-create-auto-scaling-group` and specify the following parameters:
 - Auto Scaling group name: `my-vpc-asg`
 - Launch configuration name: `my-vpc-lc`
 - Availability Zone: `"us-east-1a"`
 - Minimum size: 1
 - Maximum size: 1
 - Desired capacity: 1
 - VPC zone identifier: `"subnet-530fc83a"`

Important

Do NOT use the VPC identifier to specify the VPC zone identifier parameter. A VPC has both a VPC identifier (e.g., vpc-1a2b3c4d) and a subnet identifier (e.g., subnet-9d4a7b6c). You must use the subnet identifier instead of the VPC identifier.

Note

The Availability Zone of the subnet must match the Availability Zone you specify. In addition, when you specify several subnets and Availability Zones separated by commas (,) make sure that there is no space between the comma and the next item.

- Region: us-east-1

Your command should look similar to the following example:

```
as-create-auto-scaling-group my-vpc-asg --launch-configuration my-vpc-lc -  
-availability-zones  
"us-east-1a" --min-size 1 --max-size 1 --desired-capacity 1 --vpc-zone-  
identifier "subnet-530fc83a"
```

2. You get a confirmation that your Auto Scaling group was created successfully.

```
OK-Created AutoScalingGroup
```

Confirm that Instances Launched in Subnets

After you have created your Auto Scaling group in the VPC, verify that your Auto Scaling group has launched instance in the VPC subnet that you specified.

This section walks you through the steps for verifying that your Auto Scaling group has launched instances in the VPC using either the AWS Management Console or the Auto Scaling command line interface (CLI).

Topics

- [Confirm Instances Using the AWS Management Console \(p. 136\)](#)
- [Confirm Instances Using the Command Line Interface \(p. 137\)](#)

Confirm Instances Using the AWS Management Console

To verify that your Auto Scaling launched instances in the VPC

1. On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.
2. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab.
3. Click the **Scaling History** tab. The **Status** column lets you know the current status of your instance. When your instance is launching, the status column shows `Not yet in service`. 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. In the bottom pane, click the **Instances** tab.
5. On the **Instances** view pane, you can view the current **Lifecycle** state of your newly launched instance.

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

Confirm Instances Using the Command Line Interface

To verify that your Auto Scaling launched instance in the VPC

1. Use the `as-describe-auto-scaling-groups` command and specify the following parameter:

- Auto Scaling group name = `my-vpc-asg`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-vpc-asg --headers
```

2. The response includes the details of the specified Auto Scaling group:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES		
	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY		
AUTO-SCALING-GROUP	my-vpc-asg	my-vpc-lc	us-east-1a 1 1		
1					
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG
INSTANCE	i-95f4c3c0	us-east-1a	InService	Healthy	my-vpc-lc

Your Auto Scaling group has launched one instance in the VPC.

3. Use the `as-describe-scaling-activities` command as in the following example to list the scaling activities for the Auto Scaling group.

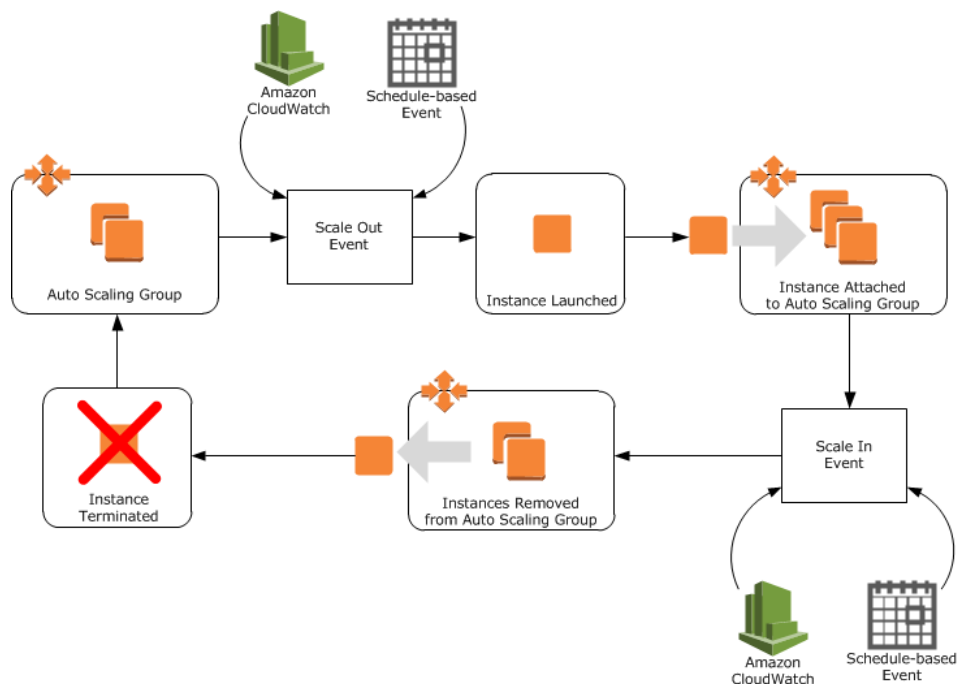
```
as-describe-scaling-activities --auto-scaling-group my-vpc-asg --headers
```

The response shows one activity that was successfully run for Auto Scaling group `my-vpc-asg`.

ACTIVITY	ACTIVITY-ID	END-TIME	GROUP-
NAME	CODE		
ACTIVITY	188b5c5f-79fc-4dcc-95a3-2762eba64702	2011-12-12T22:46:08Z	my-
vpc-asg	Successful		

Controlling How Instances Launch and Terminate

The section, [Auto Scaling Group Lifecycle](#) (p. 7), describes the basic lifecycle of instances as they launch or terminate within an Auto Scaling group.

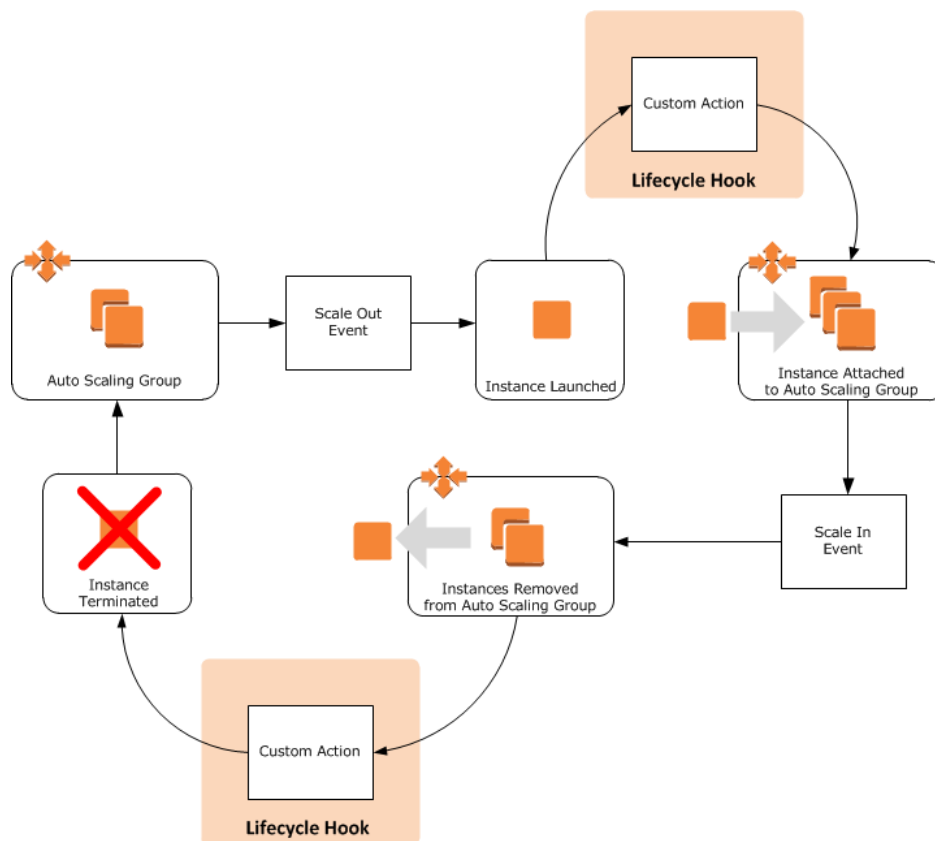


As this diagram illustrates, the standard procedure is for Auto Scaling to launch and configure an instance in response to a scale out event. When the instance is ready, it is immediately put into service. The same is true when a scale in event occurs. Auto Scaling selects an instances (based on any existing [termination policies](#) (p. 49) 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 to have more granular control over when instances are put into service and when they terminate. The following sections describe how you can implement this level of control through the use of [lifecycle hooks](#) (p. 138), and provide examples of how you might put lifecycle hooks to use.

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 remains in a wait state until you manually instruct Auto Scaling to continue or when the timeout value for the lifecycle hook expires.

Note

You can only add lifecycle hooks to your Auto Scaling group through the Auto Scaling CLI or API. You cannot add a lifecycle hook using the AWS Management Console.

The following sections can help you understand how to add lifecycle hooks to your Auto Scaling groups:

- [Adding lifecycle hooks \(p. 139\)](#)
- [Considerations when using lifecycle hooks \(p. 141\)](#)

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. To see how many lifecycle hooks you can have per account, see [Auto Scaling Account Limits](#).

To add a lifecycle hook to an Auto Scaling group

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an AWS Identity and Access Management role.

For more information, see [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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action on the instance before it becomes an active part of your application. You create a lifecycle hook with the [put-lifecycle-hook](#) command. Here is an example:

```
aws autoscaling put-lifecycle-hook my-lifecycle-hook --auto-scaling-group my-asg --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING --notification-role arn:aws:iam::896650972448:role/AutoScaling --notification-target arn:aws:sns:us-west-2:856294301421:myTopic
```

At this point, you now have a lifecycle hook attached to your Auto Scaling group. When a scale out event occurs, the Auto Scaling group launches an instance. If the Auto Scaling group uses a configuration script, that script is also 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 for you to perform a custom action.

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 did not get configured correctly and terminates it. To change these default behaviors, see the help included with the [aws autoscaling put-lifecycle-hook](#).

You can keep an instance in the `Pending:Wait` state longer by using the [aws autoscaling record-lifecycle-action-heartbeat](#) command, as shown in the next step.

4. Perform a custom action.

Auto Scaling uses a parameter, `--heartbeat-timeout`, to determine how long it should keep an instance in the `Pending:Wait` state. If you find that you need more time, you can use the [aws autoscaling record-lifecycle-action-heartbeat](#) command to restart the heartbeat timeout and keep the instance in the `Pending:Wait` state. This command requires the lifecycle action token, which was included in the message sent to your notification target.

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

Again, this command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to complete your software installation, so you use the [aws autoscaling record-lifecycle-action-heartbeat](#). This command restarts the timeout value, giving you a total of 90 minutes to complete the software installation.

5. Notify Auto Scaling that the software is installed.

When you finish installing the additional software, you need to let Auto Scaling know it can proceed with adding the instance to your Auto Scaling group. You accomplish this with the [aws autoscaling complete-lifecycle-action](#) command. Here is an example:

```
aws autoscaling complete-lifecycle-action bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
--lifecycle-hook my-lifecycle-hook --auto-scaling-group 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 the group continues to perform as expected.

Topics

- [Keeping Instances in a Wait State \(p. 141\)](#)
- [Cooldowns and custom actions \(p. 141\)](#)
- [Abandon or Continue \(p. 142\)](#)
- [Spot Instances \(p. 142\)](#)

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 a few ways:

- Changing 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.
- Calling the [aws autoscaling complete-lifecycle-action](#) or `CompleteLifecycleAction` command, which tells Auto Scaling that the instance is ready to continue on to the next state.
- Calling the [aws autoscaling record-lifecycle-action-heartbeat](#) or `RecordLifecycleActionHeartbeat`. This command increments 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 the `RecordLifecycleActionHeartbeat` command.

Cooldowns and custom actions

Each time Auto Scaling launches or terminates an instance, a [cooldown \(p. 46\)](#) takes effect. This cooldown helps ensure that the Auto Scaling group does not launch or terminate more instances than needed.

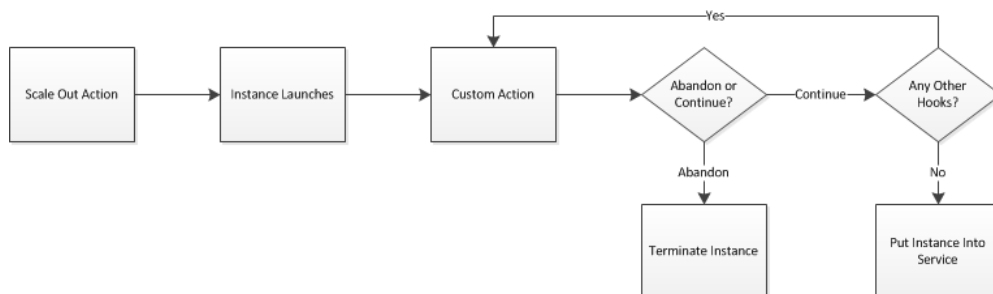
When you put a lifecycle hook on an Auto Scaling group, any scaling actions are suspended until the instance move out of a Wait state. After the instance moves out of a Wait state, the cooldown period starts.

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.

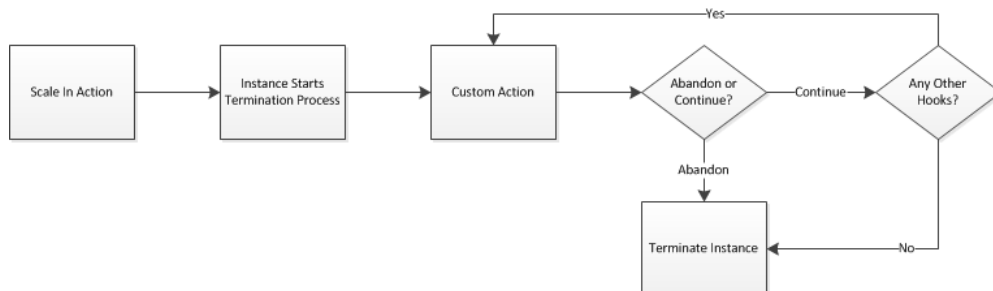
Abandon or Continue

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 (such as with the **as-complete-lifecycle-action** command or **CompleteLifecycleAction** API call).

For more information, see [Spot Instances](#).

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.

Topics

- [Installing Software to Pending Instances](#) (p. 143)
- [Filling a Cache of Servers](#) (p. 147)
- [Analyzing an Instance Before Termination](#) (p. 151)
- [Retrieving Logs from Terminating Instances](#) (p. 154)

Installing Software to Pending Instances

As with a standalone EC2 instance, you have the option of configuring instances launched into an Auto Scaling group. You configure instances using the `UserData` field in the AWS Management Console, or through the `--userdata` parameter in the Auto Scaling CLI or AWS CLI. These options help ensure that instances added to your Auto Scaling group are ready to become functional resources of your application. While an instance is configured, it remains in the `Pending` state.

In some situations, you may have software that you cannot install through a configuration script, or that you need to modify manually before Auto Scaling adds the instance to the group. You can accomplish this by adding 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` state while you install and configure the additional software.

The following steps demonstrate how to add a lifecycle hook to your Auto Scaling group.

Adding Software Using the Command Line Interface

The following steps demonstrate the general process for using the Auto Scaling CLI to install additional software to instances joining an Auto Scaling group. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information on creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To add software manually to pending instances using the CLI

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an AWS Identity and Access Management role.

For more information, see [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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, installing additional software) on the instance before it becomes an active part of your application. You create a lifecycle hook with the **as-put-lifecycle-hook** command. Here is an example:

```
as-put-lifecycle-hook ReadyForSoftwareInstall --auto-scaling-group my-asg  
--lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING --notification-  
role arn:aws:iam::896650972448:role/AutoScaling --notification-target  
arn:aws:sns:us-west-2:856294301421:myTopic
```

At this point, you now have a lifecycle hook attached to your Auto Scaling group. When a scale out event occurs, the Auto Scaling group launches an instance. If the Auto Scaling group uses a configuration script, that script is also 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 for you to install your 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 did not get configured correctly and terminates it. To change these default behaviors, see the help included with the **as-put-lifecycle-hook**.

You can keep an instance in the `Pending:Wait` state longer by using the **as-record-lifecycle-action-heartbeat** command, as shown in the next step.

4. Install the software.

Auto Scaling uses a parameter, `--heartbeat-timeout`, to determine how long it should keep an instance in the `Pending:Wait` state. If you find that you need more time, you can use the **as-record-lifecycle-action-heartbeat** command to restart the heartbeat timeout and keep the instance in the `Pending:Wait` state. This command requires the lifecycle action token, which was included in the message sent to your notification target.

```
as-record-lifecycle-action-heartbeat bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635  
--auto-scaling-group my-asg --lifecycle-hook ReadyForSoftwareInstall
```

Again, this command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to complete your software installation, so you use the **as-record-lifecycle-action-heartbeat**. This command restarts the timeout value, giving you a total of 90 minutes to complete the software installation.

5. Notify Auto Scaling that the software is installed.

When you finish installing the additional software, you need to let Auto Scaling know it can proceed with adding the instance to your Auto Scaling group. You accomplish this with the **as-complete-lifecycle-action** command. Here is an example:

```
as-complete-lifecycle-action bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-  
hook ReadyForSoftwareInstall --auto-scaling-group my-asg --lifecycle-action-  
result CONTINUE
```

Adding Software Using the Query API

The following steps demonstrate the general process for using the Query API to install additional software to instances joining an Auto Scaling group. This example assumes that you already have an Auto Scaling group, *my-asg*, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To add software manually to pending instances using the Query API

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an IAM role.

For more information, see [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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, installing additional software) on the instance before it becomes an active part of your application. You create a lifecycle hook with the **PutLifecycleHook**. Here is an example:

```
http://autoscaling.amazonaws.com/?RoleARN=arn%3Aaws%3Ai
am%3A%3A896650972448%3Arole%2FAutoScaling&AutoScalingGroupName=my-asg&Life
cycleHookName=ReadyForSoftwareInstall
&NotificationTargetARN=arn%3Aaws%3Asqs%3Aus-east-1%3A896650972448%3Alife
cyclehookqueue&LifecycleTransition=autoscaling%3AEC2_INSTANCE_LAUNCHING
&Version=2011-01-01&Action=PutLifecycleHook&SignatureVersion=2&SignatureMeth
od=HmacSHA256&Timestamp=2014-06-19T15%3A27%3A04.705Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<PutLifecycleHookResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
01-01/">
  <PutLifecycleHookResult/>
  <ResponseMetadata>
    <RequestId>2b18af33-f7c6-11e3-92e1-372b5dcd5c88</RequestId>
  </ResponseMetadata>
</PutLifecycleHookResponse>
```

At this point, you now have a lifecycle hook attached to your Auto Scaling group. When a scale out event occurs, the Auto Scaling group launches an instance. If the Auto Scaling group uses a configuration script, that script is also applied. When these steps are complete, the lifecycle hook puts the instance into the `Pending:Wait` state and uses your notification target to inform you that the instance is ready for you to install your 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 did not get configured correctly and terminates it. To change these default behaviors, see [PutLifecycleHook](#).

You can keep an instance in the `Pending:Wait` state longer by using the **RecordLifecycleAction-Heartbeat** command, as shown in the next step.

4. Install the software.

Auto Scaling uses a parameter, `HeartbeatTimeout`, to determine how long it should keep an instance in the `Pending:Wait` state. If you find that you need more time, you can use the `RecordLi-`

fecycleActionHeartbeat to restart the heartbeat timeout and keep the instance in the Pending:Wait state. This command requires the lifecycle action token, which was included in the message sent to your notification target.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionToken=d72cad36-9a9d-48c2-ac9b-f83135b9723a&LifecycleHookName=ReadyForSoftwareInstall
&Version=2011-01-01&Action=RecordLifecycleActionHeartbeat&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A33%3A41.335Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<RecordLifecycleActionHeartbeatResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <RecordLifecycleActionHeartbeatResult/>
  <ResponseMetadata>
    <RequestId>l77ae78d-f7c7-11e3-a516-d7dba8f8d058</RequestId>
  </ResponseMetadata>
</RecordLifecycleActionHeartbeatResponse>
```

Again, this command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to complete your software installation, so you use the **RecordLifecycleActionHeartbeat**. This command restarts the timeout value, giving you a total of 90 minutes to complete the software installation.

5. Notify Auto Scaling that the software is installed.

When you finish installing the additional software, you need to let Auto Scaling know it can proceed with adding the instance to your Auto Scaling group. You accomplish this with the **CompleteLifecycleAction** command. Here is an example:

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionResult=CONTINUE&LifecycleActionToken=bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
&LifecycleHookName=ReadyForSoftwareInstall&Version=2011-01-01&Action=CompleteLifecycleAction&SignatureVersion=2&SignatureMethod=HmacSHA256
&Timestamp=2014-06-19T22%3A09%3A09.174Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<CompleteLifecycleActionResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <CompleteLifecycleActionResult/>
  <ResponseMetadata>
    <RequestId>565a7789-f7fe-11e3-8d98-cdb17f635ff6</RequestId>
  </ResponseMetadata>
</CompleteLifecycleActionResponse>
```

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 minimize any decreases in application responsiveness, which in turn can provide a better customer experience.

The following steps demonstrate how to fill a cache of servers for an Auto Scaling group.

Filling a Cache Using the Command Line Interface

The following steps demonstrate how to use the Auto Scaling CLI to create a cache of servers for your Auto Scaling group. This example assumes that you already have an Auto Scaling group, *my-asg*, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To fill a cache of servers using the CLI

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an IAM role.

For more information, [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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, waiting until a specific period of time elapses) on the instance before it becomes an active part of your application. You create a lifecycle hook with the **as-put-lifecycle-hook**. Here is an example:

```
as-put-lifecycle-hook CachedServers --auto-scaling-group my-asg --lifecycle-  
transition autoscaling:EC2_INSTANCE_LAUNCHING --notification-role  
arn:aws:iam::896650972448:role/AutoScaling --notification-target  
arn:aws:sns:us-west-2:856294301421:myTopic
```

At this point, you now have a lifecycle hook that applies to any new instances launched for the Auto Scaling group. When a scale out event occurs, the Auto Scaling group launches an instance. If the Auto Scaling group uses a configuration script, that script is also applied. When these steps are complete, the lifecycle hook puts the instance into the `Pending:Wait` and uses your notification target to inform you that the instance is ready to be added to the Auto Scaling group.

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 did not get configured correctly and terminates it. To change these default behaviors, see the help included with the **as-put-lifecycle-hook**.

4. Increase the size of your Auto Scaling group.

At this point, you can increase the size of your Auto Scaling group. The new instances launched will remain in the `Pending:Wait` state until you are ready for them to be put into service.

You can increase the size of your Auto Scaling group using a command such as the following:

```
as-update-auto-scaling-group my-asg --max-size 10 --desired-capacity 8
```

The command in the preceding example increases the maximum size of the Auto Scaling group to 10 and the desired capacity for the group to 8.

5. Keep the instances in the `Pending:Wait` state until you are ready to put them into service.

To keep the instances in the `Pending:Wait` state, use the `as-record-lifecycle-action-heartbeat` command. This command requires the lifecycle action token, which was included in the message sent to your notification target.

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

This command resets the timeout value for the instance. For example, consider a lifecycle hook uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need to wait before you put these instances into service, so you use the **as-record-lifecycle-action-heartbeat**. This command restarts the timeout value, keeping the instances in a wait state for a total of 90 minutes.

6. Put the instances into service.

When you are ready for the instances to be put in service, you need to let Auto Scaling know it can proceed with adding the instance to your application. You accomplish this with the **as-complete-lifecycle-action** command. Here is an example:

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

Filling a Cache Using the Query API

The following steps demonstrates how to use the Auto Scaling Query API to create a cache of servers for your Auto Scaling group. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To fill a cache of servers using the Query API

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an AWS Identity and Access Management role.

To create an IAM role, follow 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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, waiting until a specific period of time elapses) on the instance before it becomes an active part of your application. You create a lifecycle hook with the **PutLifecycleHook**. Here is an example:

```
http://autoscaling.amazonaws.com/?RoleARN=arn%3Aaws%3Aiam%3A%3A896650972448%3Arole%2FAutoScaling&AutoScalingGroupName=my-asg&LifecycleHookName=CachedServers
&NotificationTargetARN=arn%3Aaws%3Asqs%3Aus-east-1%3A896650972448%3Alifecyclehookqueue&LifecycleTransition=autoscaling%3AEC2_INSTANCE_LAUNCHING
&Version=2011-01-01&Action=PutLifecycleHook&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A27%3A04.705Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<PutLifecycleHookResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutLifecycleHookResult/>
  <ResponseMetadata>
    <RequestId>2b18af33-f7c6-11e3-92e1-372b5dcd5c88</RequestId>
  </ResponseMetadata>
</PutLifecycleHookResponse>
```

At this point, you now have a lifecycle hook that applies to any new instances launched for the Auto Scaling group. When a scale out event occurs, the Auto Scaling group launches an instance. If the Auto Scaling group uses a configuration script, that script is also applied. When these steps are complete, the lifecycle hook puts the instance into the `Pending:Wait` and uses your notification target to inform you that the instance is ready to be added to the Auto Scaling group.

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 did not get configured correctly and terminates it. To change these default behaviors, see [PutLifecycleHook](#).

4. Increase the size of your Auto Scaling group.

At this point, you can increase the size of your Auto Scaling group. The new instances launched remain in the `Pending:Wait` state until you are ready for them to be put in service.

You can increase the size of your Auto Scaling group using a command such as the following:

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&MaxSize=10&DesiredCapacity=8&Version=2011-01-01&Action=UpdateAutoScalingGroup&SignatureVersion=2
&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T22%3A36%3A42.188Z&AUTHPARAMS
```

The command in the preceding example increases the maximum size of the Auto Scaling group to 10 and the desired capacity for the group to 8. When you use this command, Auto Scaling returns a response similar to the following:

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
```

```
<ResponseMetadata>
  <RequestId>2fa4a866-f802-11e3-8d98-cdb17f635ff6</RequestId>
</ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

5. Keep the instances in the `Pending:Wait` state until you are ready to put them into service.

To keep the instances in the `Pending:Wait` state, use the `RecordLifecycleHeartbeat` command. This command requires the lifecycle action token, which is sent to your notification target.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionToken=d72cad36-9a9d-48c2-ac9b-f83135b9723a&LifecycleHookName=CachedServers&Version=2011-01-01&Action=RecordLifecycleActionHeartbeat&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A33%3A41.335Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<RecordLifecycleActionHeartbeatResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <RecordLifecycleActionHeartbeatResult/>
  <ResponseMetadata>
    <RequestId>177ae78d-f7c7-11e3-a516-d7dba8f8d058</RequestId>
  </ResponseMetadata>
</RecordLifecycleActionHeartbeatResponse>
```

This command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need to wait before you put these instances into service, so you use the **RecordLifecycleActionHeartbeat**. This command restarts the timeout value, keeping the instances in a wait state for a total of 90 minutes.

6. Put the instances into service.

When you are ready for the instances to be put in service, you need to let Auto Scaling know it can proceed with adding the instance to your application. You accomplish this with the **CompleteLifecycleAction** command. Here is an example:

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionResult=CONTINUE&LifecycleActionToken=bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635&LifecycleHookName=ReadyForSoftwareInstall&Version=2011-01-01&Action=CompleteLifecycleAction&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T22%3A09%3A09.174Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<CompleteLifecycleActionResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <CompleteLifecycleActionResult/>
  <ResponseMetadata>
    <RequestId>565a7789-f7fe-11e3-8d98-cdb17f635ff6</RequestId>
  </ResponseMetadata>
</CompleteLifecycleActionResponse>
```

```
</ResponseMetadata>  
</CompleteLifecycleActionResponse>
```

Analyzing an Instance Before Termination

A primary benefit to 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 may want to understand why an instance is being terminated so 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, giving you time to connect to the instance and investigate the cause of the termination. The instance remains in this state until its status is set to `Terminating:Proceed`.

Note

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 want to put the instance in service, you need to put the instance `Standby` state before the termination process starts. For more information, see [Troubleshooting Instances in an Auto Scaling Group](#) (p. 217).

Analyzing Instances Using the Command Line Interface

The following steps demonstrate the general process for using the CLI to put instances in a `Terminating:Wait` state to connect to the instance and analyze what may have happened to cause the instance to fail. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To access an Auto Scaling instance before it terminates using the CLI

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an AWS Identity and Access Management role.

To create an IAM role, follow 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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, analyze the instance) on the instance before it terminates. You create a lifecycle hook with the **as-put-lifecycle-hook**. Here is an example:

```
as-put-lifecycle-hook WaitForDiagnostics --auto-scaling-group my-asg --li  
fecycle-transition autoscaling:EC2_INSTANCE_TERMINATING --notification-role  
arn:aws:iam::896650972448:role/AutoScaling --notification-target  
arn:aws:sns:us-west-2:856294301421:myTopic
```

At this point, you now have a lifecycle hook that applies to any instances that are being terminated from the Auto Scaling group. When a scale in event occurs, the Auto Scaling group puts the instance in a `Terminating:Wait` state and uses your notification target to inform you that the instance is terminating.

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. To change these default behaviors, see the help included with the **as-put-lifecycle-hook** command.

4. Connect to the instance.

When an instance is terminated, Auto Scaling 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 on how to connect to an EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Keep the instances in a terminating state until you have all the data you need.

To keep the instances in the `Terminating:Wait` state, use the **as-record-lifecycle-action-heartbeat** command.

```
as-record-lifecycle-action-heartbeat --lifecycle-hook --auto-scaling-group  
my-asg --lifecycle-hook WaitForDiagnositics
```

This command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to analyze the instance, so you use the **as-record-lifecycle-action-heartbeat**. This command restarts the timeout value, giving you a total of 90 minutes to complete your analysis.

6. Terminate the instance.

When you are ready for the instances to terminate, you can either let the timeout value expire, or use the **CompleteLifecycleAction** command. Here is an example:

```
as-complete-lifecycle-action bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635 --lifecycle-  
hook WaitForDiagnositics --auto-scaling-group my-asg -lifecycle-action-result  
CONTINUE
```

Analyzing Instances Using the Query API

The following steps demonstrate the general process for using the Query API to put instances in a `Terminating:Wait` state to connect to the instance and analyze what may have happened to cause the instance to fail. This example assumes that you already have an Auto Scaling group, *my-asg*, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To access an Auto Scaling instance before it terminates using the Query API

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an IAM role.

To create an IAM role, follow 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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, analyze the instance) on the instance before it becomes an active part of your application. You create a lifecycle hook with the **PutLifecycleHook**. Here is an example:

```
http://autoscaling.amazonaws.com/?RoleARN=arn%3Aaws%3Aiam%3A%3A896650972448%3Arole%2FAutoScaling&AutoScalingGroupName=my-asg&LifecycleHookName=WaitForDiagnostics
&NotificationTargetARN=arn%3Aaws%3Asqs%3Aus-east-1%3A896650972448%3Alifecyclehookqueue&LifecycleTransition=autoscaling%3AEC2_INSTANCE_LAUNCHING
&Version=2011-01-01&Action=PutLifecycleHook&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A27%3A04.705Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<PutLifecycleHookResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutLifecycleHookResult/>
  <ResponseMetadata>
    <RequestId>2b18af33-f7c6-11e3-92e1-372b5dcd5c88</RequestId>
  </ResponseMetadata>
</PutLifecycleHookResponse>
```

At this point, you now have a lifecycle hook that applies to any instances that are being terminated from the Auto Scaling group. When a scale in event occurs, the Auto Scaling group puts the instance in a `Terminating:Wait` state and uses your notification target to inform you that the instance is terminating.

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. To change these default behaviors, see [PutLifecycleHook](#).

4. Connect to the instance.

When an instance is terminated, Auto Scaling 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 on how to connect to an EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Keep the services in a terminating state until you have all the data you need.

To keep the services in the `Terminating:Wait` state, use the `RecordLifecycleActionHeartbeat` command.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionToken=d72cad36-9a9d-48c2-ac9b-f83135b9723a&LifecycleHookName=WaitForDiagnostics
&Version=2011-01-01&Action=RecordLifecycleActionHeartbeat&SignatureVer
```

```
sion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A33%3A41.335Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<RecordLifecycleActionHeartbeatResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <RecordLifecycleActionHeartbeatResult/>
  <ResponseMetadata>
    <RequestId>177ae78d-f7c7-11e3-a516-d7dba8f8d058</RequestId>
  </ResponseMetadata>
</RecordLifecycleActionHeartbeatResponse>
```

This command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to analyze the instance, so you use the **RecordLifecycleHeartbeat**. This command restarts the timeout value, giving you a total of 90 minutes to complete your analysis.

6. Terminate the instance.

When you are ready for the instances to terminate, you can either let the timeout value expire, or use the **CompleteLifecycleAction** command. Here is an example:

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleAc
tionResult=CONTINUE&LifecycleActionToken=bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
&LifecycleHookName=WaitForDiagnostics&Version=2011-01-01&Action=CompleteLi
fecycleAction&SignatureVersion=2&SignatureMethod=HmacSHA256
&Timestamp=2014-06-19T22%3A09%3A09.174Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<CompleteLifecycleActionResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <CompleteLifecycleActionResult/>
  <ResponseMetadata>
    <RequestId>565a7789-f7fe-11e3-8d98-cdb17f635ff6</RequestId>
  </ResponseMetadata>
</CompleteLifecycleActionResponse>
```

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 `Terminating` state. The instance remains in this state until the instance fully terminates.

By default, you cannot connect to an instance that is terminating. However, you can choose to put the instance in a `Terminating:Wait` state using the **PutLifecycleHook** command. This option allows you to access the terminating instance. This can be useful when you want to retrieve log files from the instance for later analysis.

Note

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 want to put the instance in service, you need to put the instance in the `Standby` state before the termination process starts. For more information, see [Troubleshooting Instances in an Auto Scaling Group](#) (p. 217).

Retrieving Logs Using the Command Line Interface

The following steps demonstrate the general process for using the CLI to put instances in a `Terminating:Wait` state to connect to the instance to download log files. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information on creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To retrieve logs from a terminating server using the CLI

1. Create a notification target.

This target receives the notification that an instance is in the `Pending:Waiting` state. You can either [create a topic using Amazon SNS](#), or use an [Amazon SQS queue](#).

When you create your target, make a note of its Amazon Resource Name (ARN). You will use it in the next step.

2. Create an IAM role.

To create an IAM role, follow 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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, retrieve log files) on the instance before it terminates. You create a lifecycle hook with the **as-put-lifecycle-hook**. Here is an example:

```
as-put-lifecycle-hook GetLogs --auto-scaling-group my-asg --lifecycle-  
transition autoscaling:EC2_INSTANCE_TERMINATING --notification-role  
arn:aws:iam::896650972448:role/AutoScaling --notification-target  
arn:aws:sns:us-west-2:856294301421:myTopic
```

At this point, you now have a lifecycle hook that applies to any instances that are being terminated from the Auto Scaling group. When a scale in event occurs, the Auto Scaling group puts the instance in a `Terminating:Wait` state and uses your notification target to inform you that the instance is terminating.

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. To change these default behaviors, see the help included with the **as-put-lifecycle-hook** command.

4. Connect to the instance.

When an instance is terminated, Auto Scaling sends a message to the notification target that you created. After you receive this notification, you can connect to the instance to download the log files

you need. For more information about how to connect to an EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Keep the instances in a terminating state until you have all the data you need.

To keep the instances in the `Terminating:Wait` state, use the `as-record-lifecycle-action-heartbeat` command.

```
as-record-lifecycle-action-heartbeat --lifecycle-hook --auto-scaling-group my-asg --lifecycle-hook GetLogs
```

This command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to retrieve the log files, so you use the `as-record-lifecycle-action-heartbeat`. This command restarts the timeout value, giving you a total of 90 minutes to get the files.

6. Terminate the instance.

When you are ready for the instances to terminate, you can either let the timeout value expire, or use the `as-complete-lifecycle-action` command. Here is an example:

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

Retrieving Logs Using the Query API

The following steps demonstrate the general process for using the Query API to put instances in a `Terminating:Wait` state to connect to the instance to download log files. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To retrieve logs from a terminating server using the Query API

1. Create a notification target.

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

When you create your target, make a note of its Amazon Resource Name (ARN).

2. Create an AWS Identity and Access Management role.

To create an IAM role, follow 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**.

When you create your role, make a note of its ARN.

3. Create a lifecycle hook.

A lifecycle hook tells Auto Scaling that you want to perform an action (in this case, retrieve log files) on the instance before it terminates. You create a lifecycle hook with the `PutLifecycleHook`. Here is an example:

```
http://autoscaling.amazonaws.com/?RoleARN=arn%3Aaws%3Aiam%3A%3A896650972448%3Arole%2FAutoScaling&AutoScalingGroupName=my-asg&LifecycleHookName=GetLogs
&NotificationTargetARN=arn%3Aaws%3Asqs%3Aus-east-1%3A896650972448%3Alifecyclehookqueue&LifecycleTransition=autoscaling%3AEC2_INSTANCE_LAUNCHING
&Version=2011-01-01&Action=PutLifecycleHook&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A27%3A04.705Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<PutLifecycleHookResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <PutLifecycleHookResult/>
  <ResponseMetadata>
    <RequestId>2b18af33-f7c6-11e3-92e1-372b5dcd5c88</RequestId>
  </ResponseMetadata>
</PutLifecycleHookResponse>
```

At this point, you now have a lifecycle hook that applies to any instances that are being terminated from the Auto Scaling group. When a scale in event occurs, the Auto Scaling group puts the instance in a `Terminating:Wait` state.

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. To change these default behaviors, see [PutLifecycleHook](#).

4. Connect to the instance.

When an instance is terminated, Auto Scaling sends a message to the notification target that you created. After you receive this notification, you can connect to the instance to download the log files you need. For more information on how to connect to an EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Keep the instances in a terminating state until you have all the data you need.

To keep the instances in the `Terminating:Wait` state, use the `RecordLifecycleActionHeartbeat` command.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionToken=d72cad36-9a9d-48c2-ac9b-f83135b9723a&LifecycleHookName=GetLogs
&Version=2011-01-01&Action=RecordLifecycleActionHeartbeat&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T15%3A33%3A41.335Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<RecordLifecycleActionHeartbeatResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <RecordLifecycleActionHeartbeatResult/>
  <ResponseMetadata>
    <RequestId>177ae78d-f7c7-11e3-a516-d7dba8f8d058</RequestId>
  </ResponseMetadata>
</RecordLifecycleActionHeartbeatResponse>
```

This command resets the timeout value for the instance. For example, consider a lifecycle hook that uses the default timeout value of 60 minutes. After 30 minutes, you discover that you need more time to retrieve the log files, so you use the **RecordLifecycleHeartbeat**. This command restarts the timeout value, giving you a total of 90 minutes to download the files.

6. Terminate the instance.

When you are ready for the instances to terminate, you can either let the timeout value expire, or use the **CompleteLifecycleAction** command. Here is an example:

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&LifecycleActionResult=CONTINUE&LifecycleActionToken=bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635&LifecycleHookName=GetLogs&Version=2011-01-01&Action=CompleteLifecycleAction&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-19T22%3A09%3A09.174Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<CompleteLifecycleActionResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <CompleteLifecycleActionResult/>
  <ResponseMetadata>
    <RequestId>565a7789-f7fe-11e3-8d98-cdb17f635ff6</RequestId>
  </ResponseMetadata>
</CompleteLifecycleActionResponse>
```

Add, Modify, or Remove Auto Scaling Group Tags

You can organize and manage your Auto Scaling groups by assigning your own metadata to each group in the form of *tags*. You define a *key* and a *value* for each tag. The key can be a general category, such as project, owner, or environment. The value can be a specific instance in that category. For example, if one of your projects is named LIMA, you could define a tag key as "project" with a tag value "LIMA". This indicates that the Auto Scaling group is assigned to the LIMA project. Similarly, if you want to differentiate between your development environments, you could define tag with a key of "environment" and a value of "test" and another tag with a key of "environment" and a value of "production". These tags indicate that the Auto Scaling group is designated as either a test environment or a production environment. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with your Auto Scaling groups.

Optionally, you can propagate Auto Scaling group tags to the EC2 instances in the group. You can use the EC2 instance tags like any other AWS resource tags, including to show instance cost allocation by organizing your AWS bill. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of running your Auto Scaling instances, organize your billing information according to Auto Scaling instances with the same tag key values. For example, you can track the cost of running Auto Scaling instances for project LIMA in a test environment. For more information about cost allocation, see [Use Cost Allocation Tags](#) in the *AWS Billing and Cost Management User Guide*.

Tag Restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource— 10.
- Maximum key length— 127 Unicode characters.
- Maximum value length— 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.

Note

When you launch instances in an Auto Scaling group, Auto Scaling automatically tags each one with the group name. This tag can be identified by its key, `aws:autoscaling:groupName`. Tags containing the prefix `aws:` have been created by AWS. These tags cannot be edited or deleted, and they do not count toward your limit of 10 tags per 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 \(p. 31\)](#).

This section walks you through the process for adding, modifying, or deleting tags for your Auto Scaling group.

Topics

Add or Modify Tags for Your Auto Scaling Group

When you add tags to your Auto Scaling group, you can specify that the tags should be propagated to instances in your Auto Scaling group. These tags will be applied to new EC2 instances launched by the Auto Scaling group. Likewise, after you modify a tag, the updated version is applied to new instances launched by the Auto Scaling group after the change. Tags created or modified for an existing Auto Scaling group do not apply to instances that are already running at the time you added or modified the tags.

You can add or modify tags for your Auto Scaling group using the AWS Management Console, the Auto Scaling command line interface (CLI), the AWS CLI, the Query API, or the SDKs. This section covers instructions for adding or modifying tags using the AWS Management Console or the Auto Scaling CLI.

Topics

- [Manage Tags Using the AWS Management Console \(p. 159\)](#)
- [Manage Tags Using the Command Line Interface \(p. 160\)](#)

Manage Tags Using the AWS Management Console

Use the Amazon EC2 console to add or modify tags.

To add or modify tags

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.

3. On the Auto Scaling groups page, select your Auto Scaling group (*my-first-asg*) from the list.
4. The bottom pane displays the details of your Auto Scaling group. In the bottom pane, click the **Tags** tab.
5. In the tags pane, click **Add/Edit Tags**.
6. The **Add/Edit Auto Scaling Group Tags** wizard lists the tags that are already assigned to the Auto Scaling group. Use this wizard to modify current tags or add new ones.
7. Click the **Key** and the **Value** fields and modify the existing tag.
8. Click **Add Tag** to add a new tag.
 - a. In the **Key** and **Value** fields, enter the key and the value for your tags.
 - b. Keep the **Tag New Instances** field selected if you want to propagate the tags to the instances launched by your Auto Scaling group.

Click the **Tag New Instances** field if you do not want to propagate the tags 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.
9. When you have completed adding or modifying your tags, click **Save**.
10. A list of tags currently associated with your Auto Scaling group appears in the bottom pane. Verify your changes.

Manage Tags Using the Command Line Interface

This section walks you through the process for creating or modifying tags.

Use the `as-create-or-update-tags` command to create a tag or modify existing tags for an Auto Scaling group. The command takes the following arguments:

```
as-create-or-update-tags --tag "id=value,t=value, k=value, [v=value], [p=value]"  
[--tag "id=value, t=value, k=value, [v=value], [p=value]]" ...] [General Options]
```

Provide these values for the following tag options:

Option	Description	Example value
k	Tag key, as part of key-value pair. Always required.	environment
v	Tag value, as part of key-value pair. Optional.	test
t	The type of resource to which the tag is applied. Currently, Auto Scaling supports the <code>auto-scaling-group</code> resource type.	auto-scaling-group
id	The name of the resource (Auto Scaling group) to which the tag is applied.	my-first-asg
p	The propagate-at-launch flag. Specify this flag only if you want the tags to be applied to newly launched EC2 instances.	true

Your command should look similar to the following example:

```
as-create-or-update-tags --tag "id=my-first-asg, t=auto-scaling-group, k=environment, v=test, p=true"
```

Note

You must specify the resource name and resource type so that Auto Scaling knows the resource to which the tag applies.

Auto Scaling responds as in the following example.

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

Use the `as-describe-tags` command to verify that the tag is created. The command takes the following arguments:

```
as-describe-tags [--filter "key1=value1,key2=value2..." [--filter "key1=value1,key2=value2..." ...]] [General Options]
```

The `key1`, `key2` filter options are the resource name, resource type, tag key, tag value, and the propagate flag that you used to define your tag. For this procedure, specify the following options:

- Tag key: `key=environment`
- Tag value: `value=test`

Your command should look like the following example:

```
as-describe-tags --filter "key=environment, value=test"
```

Auto Scaling responds as in the following example:

```
TAG my-first-asg auto-scaling-group environment test true
```

Enter the `as-describe-auto-scaling-groups` command as in the following example to confirm that the tag is applied to the Auto Scaling group `my-first-asg`.

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-first-asg --headers
```

The response includes information confirming that Auto Scaling applied the tag to the Auto Scaling group `my-first-asg`. The response should look similar to the following example:

```
AUTO-SCALING-GROUP my-first-asg my-first-lc us-east-1a 1 1 1

INSTANCE INSTANCE-ID AVAILABILITY-ZONE STATE STATUS LAUNCH-CONFIG
TAG RESOURCE-ID RESOURCE-TYPE KEY VALUE PROPAGATE-AT-LAUNCH
TAG my-first-asg auto-scaling-group environment test true
```

Delete Tags

You can delete the tags associated with your Auto Scaling group at any time. This section walks you through the process for deleting the tags using either the AWS Management Console or the Auto Scaling CLI.

Delete Tags Using the AWS Management Console

Use the Auto Scaling console to delete tags.

To delete tags

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group (*my-first-asg*) from the list.
4. The bottom pane displays the details of your Auto Scaling group. In the bottom pane, click **Tags**.
5. In the tags pane, click **Add/Edit Tags**.
6. The **Add/Edit Auto Scaling Group Tags** wizard lists the tags that are already assigned to the Auto Scaling group.

Click the delete icon next to the tags you want to delete.

7. Click **Save**.

Delete Tags Using the Command Line Interface

Use the Auto Scaling CLI command `as-delete-tags` to delete the tag. The command takes the following arguments:

```
as-delete-tags --tag "id=value,t=value, k=value, [v=value], [p=value]" [--tag "id=value, t=value, k=value, [v=value], [p=value]]" ...] [General Options]
```

Note

You only need to specify the tag key, you don't need to specify the value. If you do specify a value, and if the value is incorrect, the tag is not deleted.

Your command should look like the following example:

```
as-delete-tags --tag "id=my-first-asg,t=auto-scaling-group,k=environment"
```

Auto Scaling responds as in the following example:

```
OK
```

Launching Spot Instances in Your Auto Scaling Group

If you can be flexible about when you run your applications, or if the performance of your application scales efficiently with the addition of temporary compute resources, and you want to benefit from the cost-savings of using Spot Instances, you can set up Auto Scaling to launch Spot Instances instead of

On-Demand instances. For more information about Spot Instances, see [Spot Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Spot Instances are a way for you to purchase unused Amazon EC2 capacity. You bid on certain instances, and, as long as your bid price exceeds the current Spot Price for those instances, you get to use them at the cost of the current Spot Price.

In this section, you learn how to use Auto Scaling to launch Spot Instances.

When you use Spot Instances with Auto Scaling, you need to understand the basics of Spot Instance requests and how they interact with Auto Scaling.

- **Spot market price and your bid price.** If the market price for Spot Instances rises above your Spot bid price for a running instance, Amazon EC2 terminates your instance. This is true for all Spot Instances, whether or not you manage them using Auto Scaling. If your Spot bid price exactly matches the Spot market price, your bid may or may not be fulfilled, depending on several factors—such as available Spot Instance capacity.
- **Setting your bid price.** When you use Auto Scaling to launch Spot Instances, you set your Spot bid price in an Auto Scaling launch configuration.
- **Changing your bid price.** If you want to change your Spot bid price, you have to create a new launch configuration and associate it with your Auto Scaling group. You cannot edit launch configurations, which can serve as a record of the configuration history for running and terminated instances and their bid price.
- **New bid price and running instances.** When you change your Spot bid price by creating a new launch configuration, running instances will continue to run as long as the Spot bid price for those running instances is higher than the current Spot market price.
- **Requesting On-Demand and Spot Instances with launch configurations.** If you want to change the instances that Auto Scaling launches for an existing job, you have to create a new launch configuration and associate it with your Auto Scaling group. You cannot launch On-Demand instances with Spot Instances using the same launch configuration.
- **Maintaining your Spot Instances.** When your instance is terminated (whether it's a Spot Instance or an On-Demand instance), Auto Scaling launches another instance in an effort to replace it and maintain the desired capacity specified in the Auto Scaling group. However, whether Auto Scaling successfully launches an instance depends on the Spot bid price as compared to the Spot market price: If the bid price is higher than the market price, then an instance is launched; if the market price is higher than the bid price, then no instance is launched.
- **Auto Scaling and Spot Instance termination of instances.** Amazon EC2 terminates a Spot Instance when the bid price for that instance falls below the Spot market price. Auto Scaling terminates or replaces instances based on other criteria. For more information, see [Auto Scaling Instance Termination](#).

In this section, you create a launch configuration and an Auto Scaling group that launches Spot Instances and verify and obtain information about the new instances.

Topics

- [Launching Spot Instances in the AWS Management Console \(p. 163\)](#)
- [Launching Spot Instances Using the CLI \(p. 172\)](#)

Launching Spot Instances in the AWS Management Console

Use Auto Scaling in the AWS Management Console to create a launch configuration and an Auto Scaling group that launches Spot Instances. You can also use Auto Scaling in the AWS Management Console to verify and obtain information about the new instances. In this section, you perform the following tasks:

- [Create a Launch Configuration \(p. 164\).](#)
- [Create an Auto Scaling Group \(p. 166\).](#)
- [Verify and Check Your Instances \(p. 168\).](#)
- [Get Notifications When the Auto Scaling Group Changes \(p. 169\).](#)
- [Update the Bid Price for the Spot Instances \(p. 170\).](#)
- [Clean Up \(p. 171\).](#)

Create a Launch Configuration

In Auto Scaling, you place bids for Spot Instances using the **Create Launch Configuration** wizard. This wizard is like the **Launch Instance** wizard that you use to launch Amazon EC2 On-Demand Instances. The option to specify a Spot bid—the maximum price you are willing to pay for Spot Instances—is displayed after you select the **Request Spot Instances** purchasing option in the **Configure Details** step of this wizard.

Auto Scaling will request Spot Instances using the maximum price, but this price is not what you actually pay for the instances when they are launched. You pay the current Spot Price as long as it is lower than your bid price. For example, say you bid \$0.05 for m1.small instances. Your bid gets fulfilled if the current Spot market price for an m1.small Spot Instance is below \$0.05. If the current price is \$0.03, then you will be charged the current market price of \$0.03 per hour. In fact, as long as your Spot bid is higher than the Spot market price, your bid will be fulfilled and a Spot Instance will be launched for you. If the current Spot market price drops to \$0.02 or rises to \$0.04, you pay the new market price.

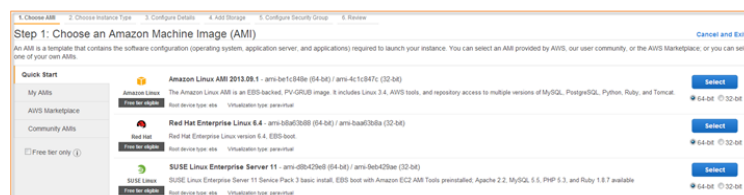
It is important that you carefully consider the price you specify for your bid. Only specify Spot bid prices that you are willing to pay. If you intentionally bid too high, you might end up paying your high bid price if the Spot market price rises. Also, if you specify a new Spot bid price that is higher than the bid price for Spot Instances that you are currently running, your new Spot bid may result in your own running instances being replaced at the higher price. To help you choose the appropriate price to bid, you can view the Spot price history using the AWS Management Console, CLI, or API. For more information, see [View Spot Price History](#) in the *Amazon EC2 User Guide for Linux Instances*.

In addition, if you want to specify a new Spot bid price, you have to create a new launch configuration, and update your Auto Scaling group to specify the new launch configuration. Launch configurations represent a record of the details of running and terminated instances. They can be helpful in tracking the history of your instances.

For more information about changing Spot bids, see [Update the Bid Price for the Spot Instances \(p. 179\)](#) later in this procedure.

To create a launch configuration

1. In the Amazon EC2 console, in the navigation pane, under Auto Scaling, click **Launch Configurations**. The **Welcome to Auto Scaling** page opens. If you have launch configurations, this page lists them.
2. Click **Create launch configuration**.
3. In the **Choose an Amazon Machine Image (AMI)** step, select an AMI to use for your instance. You can choose from your own set in **My AMIs**, or from the **Quick Start**, the **AWS Marketplace**, or the **Community AMIs** selections. In this procedure, select *Amazon Linux AMI* on the **Quick Start** page.



Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console

4. In the **Choose an Instance Type** step, select an instance type for your instances. Make your selection by looking at the complete list available (**All Instance types**), or subsets based on use categories of the instances. In this procedure, select `m1.small` from the **All Instance types** list, then click **Next: Configure details**.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Currently selected: m1.small (1 ECUs, 1 vCPUs, 1.7 GiB memory, 1 x 160 GiB Storage Capacity)

All Instance types		All instances						
Select an instance type to suit your requirements		Size	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
Micro instances	Free tier eligible	t1.micro	up to 2	1	0.613	EBS only	-	Very Low
General purpose		m1.small	1	1	1.7	1 x 160	-	Low
Memory optimized		m1.medium	2	1	3.7	1 x 410	-	Moderate
Storage optimized		m1.large	4	2	7.5	2 x 420	Yes	Moderate

For information about creating Auto Scaling launch configurations, see [Step 1: Create a Launch Configuration \(p. 29\)](#). For more information about launching EC2 instances, see [Launch Your Instance in the Amazon EC2 User Guide for Linux Instances](#).

5. In the **Create Launch Configuration** page, specify the following information:
 - Type the name of the launch configuration in the **Name** field.

Note

To help you identify which of your instances are Spot Instances, refer to their launch configurations. Consider assigning a launch configuration name that includes **Spot** and the bid price.

For this procedure, name the launch configuration `spotlc-5cents`.

1. Choose All 2. Choose Instance Type 3. Configure Details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Name

Purchasing option ☒ Request Spot Instances

IAM role

Monitoring ☐ Enable CloudWatch detailed monitoring [Learn more](#)

Advanced Details

Once created, launch configurations cannot be edited. To use a different launch configuration, create a new one and apply it to your group.

- Select the **Request Spot Instances** option.

When you select the **Request Spot Instances** option, the page expands to include information about current Spot Prices for the Availability Zone that you've selected. Use the list of current prices to determine your bid, which is the maximum price you are willing to pay for Spot Instances.

- Enter the **Maximum Price** that you're willing to pay for the Spot Instances that Auto Scaling will launch for you.

In this procedure, enter `$0.05` as your maximum price.

Auto Scaling Developer Guide

Launching Spot Instances in 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

- At this point, the rest of the steps are similar to the steps you take when you launch On-Demand Instances. You can choose to add storage and define your security groups. For information about these steps, see [Launch Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

In this procedure, skip these steps. Click **Skip to Review**.

- The **Review** step captures the parameters you specified for your Spot Instances. The details here become the template that Auto Scaling uses to launch the EC2 instances. Check the details and click **Create launch configuration**.

Note

You might get reminders about improving security for your instances and storage options.

Create Launch Configuration

AMI Details

Amazon Linux AMI 2015.09.1 - ami-83e4b0ca

Instance Type

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

Launch configuration details

Name: spot-c5c5c5c5

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

The page displays confirmation that your launch configuration was created successfully.

✓ Successfully created launch configuration: spot-c5c5c5c5

[View creation log](#)

Now that you have created your launch configuration, your next step is to create an Auto Scaling group for the Spot Instances that you want Auto Scaling to launch. The next section describes the process.

Create an Auto Scaling Group

You create your Auto Scaling group after creating the launch configuration. There are two ways to access the **Create Auto Scaling Group** wizard. If you just created a launch configuration, you can start the wizard by clicking **Create an Auto Scaling group using this launch configuration** in the page that confirms the creation of the launch configuration. You can also go to **Auto Scaling Groups** from the navigation pane in the console.

In this procedure, use the **Auto Scaling Groups** page.

Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console

When you create your Auto Scaling group, you specify the name of the launch configuration that you want Auto Scaling to use for launching your instances. Remember that the launch configuration is a template for configuration details of your instances (bid price, AMI, and instance type). With the Auto Scaling group, you specify the location of your group of instances. You also let Auto Scaling know at what level it should maintain your fleet of Spot Instances by setting the **Start with**, **minimum**, and **maximum** instances options.

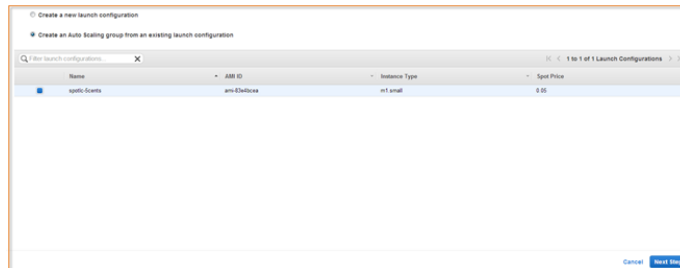
Note

The desired capacity option tells Auto Scaling the number of instances you want the service to maintain. When you use the Auto Scaling console, you set desired capacity by specifying a value for **Start with**. When you use the Auto Scaling command line interface (CLI), you set desired capacity by specifying the `desired-capacity` option in the `update-autoscaling-group` command.

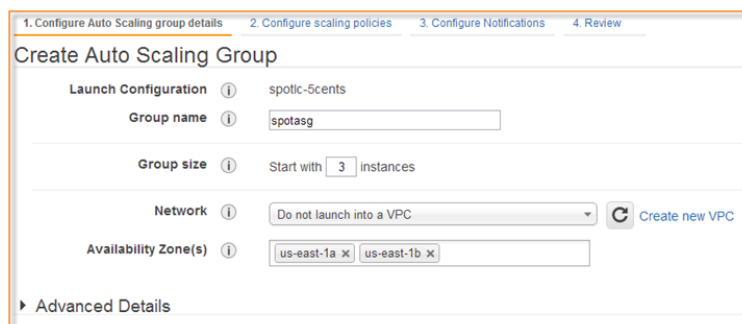
If your Spot bid price falls below the Spot Price, and Amazon EC2 terminates your instance, Auto Scaling submits your bid again and launch an instance in an effort to maintain the desired capacity you specified. However, whether Auto Scaling successfully launches an instance depends on the Spot bid price as compared to the Spot market price: If the Spot bid price is higher than the Spot market price, then an instance is launched. If the market price is higher than the bid price, then no instance is launched at that point, but Auto Scaling keeps on trying.

To create an Auto Scaling group

1. In the navigation pane, under Auto Scaling, click **Auto Scaling Groups**. The **Welcome to Auto Scaling** page opens. If you have Auto Scaling groups, this page lists them.
2. Click **Create Auto Scaling group**. The **Create Auto Scaling Group** page opens. If you have launch configurations, this page lists them. You should see the *spotlc-5cents* launch configuration that you just created. Select it and click **Next Step**.



3. In the **Configure Auto Scaling Group Details** page, specify the following information:
 - Auto Scaling Group name = `spotasg`
 - Availability Zone = `us-east-1a`, `us-east-1b`
 - Start with = 3



Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console

In this page, you can also set up load balancing, health check, and monitoring. In addition, you can continue using the wizard to configure scaling policies and notifications. However, these steps are not covered in this procedure. Click **Next: Configure scaling policies**.

4. In the **Configure scaling policies** step, select **Keep this group at its initial size**, then click **Review**.
5. The **Review** page captures the parameters you specified for your Auto Scaling group. Check the details and click **Create Auto Scaling group**.

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

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details

Group name	spotasg
Group size	3
Minimum Group Size	1
Maximum Group Size	5
Availability Zone(s)	us-east-1a, us-east-1b
Health Check Grace Period	300
Detailed Monitoring	No

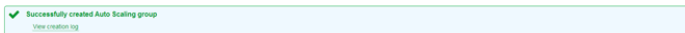
Scaling Policies

Notifications

The page displays confirmation that your bid request was submitted successfully or that Auto Scaling group was created successfully.

Note

The Spot Instances for your Auto Scaling group are launched when your Spot bid—which was specified as **Maximum price** in your launch configuration—is higher than the current Spot Price.

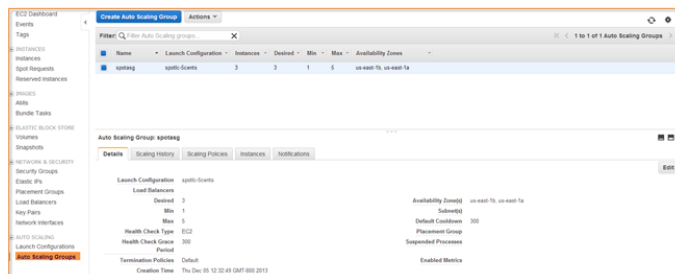


When your Auto Scaling group is created, verify the details of the group. The next section describes the process.

Verify and Check Your Instances

Use the **Auto Scaling Groups** page for information about all your account Auto Scaling groups in a region. In addition, the page contains details about specific Auto Scaling groups, the instances, and any scaling policies and notifications that have been created for the group.

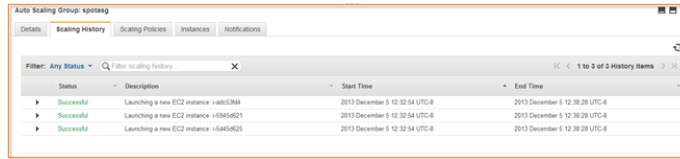
- To verify details of your new *spotasg* group, select it and in the bottom pane go to the **Details** tab.



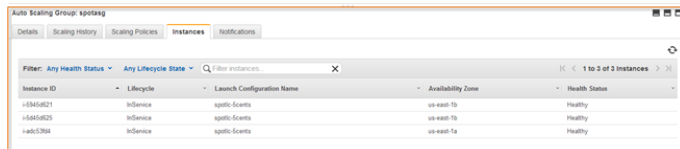
- To confirm that Auto Scaling is launching your Spot Instances according to your specifications, click the **Scaling History** tab. In this procedure, it should show that Auto Scaling successfully launched three instances.

Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console



- To get details about your Spot Instances, click the **Instances** tab. It should confirm that Auto Scaling is launching the instances you want in the Availability Zones that you specified.

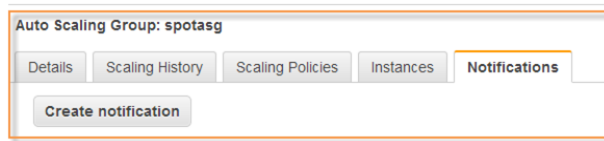


Get Notifications When the Auto Scaling Group Changes

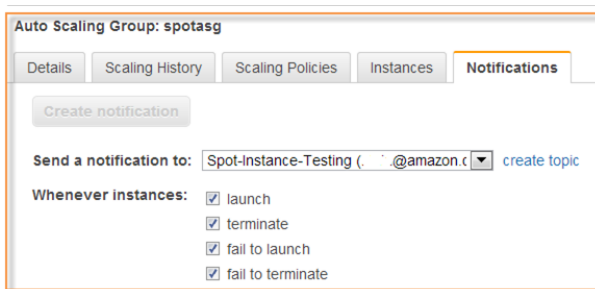
When you work with Auto Scaling in the AWS Management Console, you can set up the notification using the **Auto Scaling Groups** page.

To set up notifications

1. Select the Auto Scaling group *spotasg*, click the **Notifications** tab, and click **Create notification**.



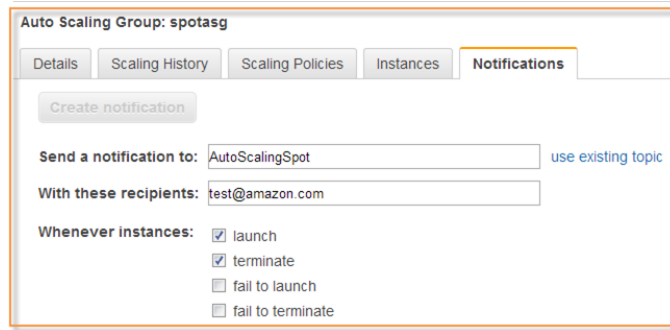
A wizard opens in the **Notifications** tab. If you have Amazon SNS topics, they are listed in the **Send a notification to** list. You may also notice that all the options under **Whenever instances** are selected. In this procedure, you create a new topic with different specifications.



2. Click **create topic**, specify the following, and click **Save**.
 - **Send a notification to** - AutoScalingSpot
 - **With these recipients** - *your email account*
 - **Whenever instances** - launch, terminate

Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console



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

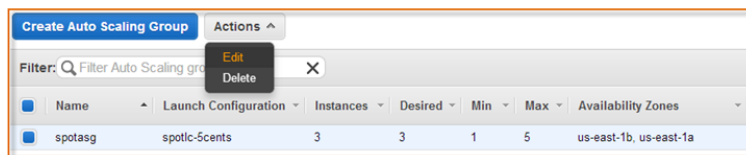
Update the Bid Price for the Spot Instances

Auto Scaling launch configurations cannot be changed. They represent a record of the launch configuration details of running and terminated instances. They can be helpful in tracking the history of your instances. If you want to modify your bid price for Spot Instances, you must create a new launch configuration.

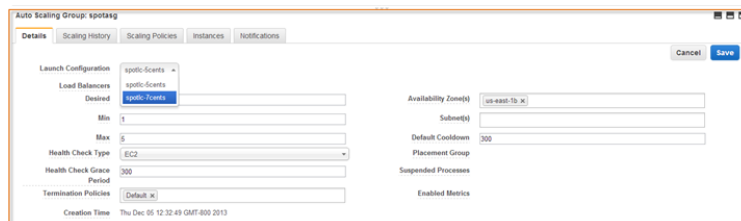
For example, if you want to launch a set of Spot Instances that have a higher likelihood of running uninterrupted for a long time, you can specify a higher Spot bid price. To do this, you must create a new launch configuration and associate it with your Auto Scaling group, using the same procedure that you followed earlier.

To update the bid price for the Spot Instances

1. Create a launch configuration with the same specifications as in [Create a Launch Configuration \(p. 164\)](#), except for the following:
 - Launch configuration name = `spotlc-7cents`
 - Spot price = \$0.07
2. On the **Auto Scaling Groups** page, select `spotasg`, click **Actions**, and then click **Edit**.



3. In the **Details** tab, specify the new launch configuration `spotlc-7cents`, change the value of **Desired** to 5, then click **Save**.

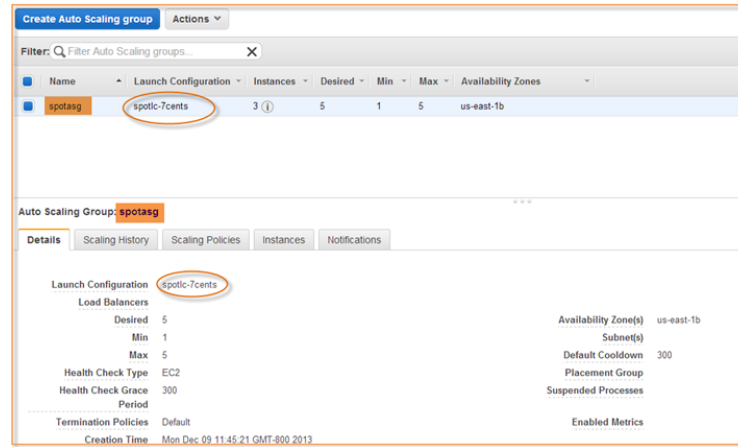


4. View your changes.

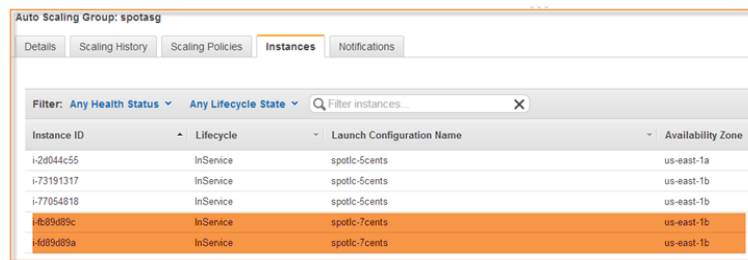
Auto Scaling Developer Guide

Launching Spot Instances in the AWS Management Console

- The **Details** tab shows that your Auto Scaling group *spotasg* is now associated with the launch configuration *spotlc-7cents* and the **Desired** value is now 5.



- The **Instances** tab shows that you have new instances launched using *spotlc-7cents*.



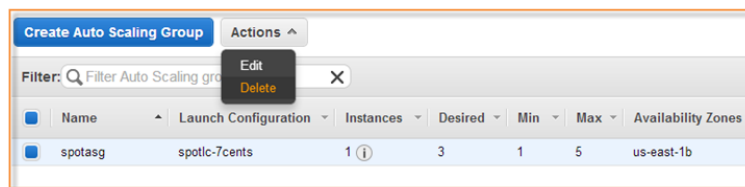
When you create a new launch configuration that sets a new bid price for Spot Instances, and you have Spot Instances already running based on a different bid price, these instances based on a different bid price continue running and are only terminated if the Spot market price goes above the bid price on which the running Spot Instance was based, or if you manually terminate them.

Clean Up

After you're finished using your instances and your Auto Scaling group, it is a good practice to clean up. This process involves deleting the Auto Scaling group, which also deletes all the Spot Instances and outstanding Spot bids that were launched as part of the group.

To clean up Auto Scaling group and instances

- Return to the **Auto Scaling Groups** page, select *spotasg*, click **Actions**, and click **Delete**.



When Amazon EC2 sends a message checking if you really want to delete your Auto Scaling group, click **Yes, Delete**.

Tasks Completed

You just performed the following tasks:

- Created a launch configuration that launched Spot Instances.
- Created an Auto Scaling group.
- Obtained information about your Auto Scaling group and instances.
- Set up notifications.
- Updated the bid price.
- Cleaned up.

Launching Spot Instances Using the CLI

Use the Auto Scaling CLI to create a launch configuration and an Auto Scaling group that launches Spot Instances. You can also use Auto Scaling in the AWS Management Console to verify the details of your launch configuration and obtain information about the new instances. In this section, you perform the following tasks:

- [Create a Launch Configuration \(p. 173\)](#), including the `--spot-price` option to specify the bid price for the Spot Instances to launch.
- [Create an Auto Scaling Group \(p. 174\)](#), specifying the launch configuration in which you specified a spot bid price.
- [Verify and Check Your Instances \(p. 175\)](#).
- [Get Notifications When the Auto Scaling Group Changes \(p. 178\)](#).
- [Set Up a Schedule for Your Spot Bids \(p. 181\)](#).
- [Clean Up \(p. 182\)](#).

Spot Instance CLI Commands

You use the following Auto Scaling CLI commands.

Command	Description
<code>as-create-launch-config</code>	Creates a new launch configuration with specified attributes.
<code>as-create-auto-scaling-group</code>	Creates a new Auto Scaling group with specified name and other attributes.
<code>as-describe-auto-scaling-groups</code>	Describes the Auto Scaling groups, if the groups exist.
<code>as-describe-auto-scaling-instances</code>	Describes the Auto Scaling instances, if the instances exist.
<code>as-describe-scaling-activities</code>	Describes a set of activities or all activities belonging to a group.

Command	Description
<code>as-put-notification-configuration</code>	Configures an Auto Scaling group to send notifications when specified events take place.
<code>as-describe-notification-configurations</code>	Returns a list of notification actions associated with Auto Scaling groups for specified events.
<code>as-put-scheduled-update-group-action</code>	Creates or updates a scheduled update group action.
<code>as-delete-auto-scaling-group</code>	Deletes the specified Auto Scaling group, if the group has no instances and no scaling activities are in progress.

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#). Use the basic scenario to get started with the infrastructure that is typically needed in Auto Scaling.

In Auto Scaling, you place bids for Spot Instances using the `as-create-launch-config` command. Specify the maximum price you are willing to pay for an instance using the `--spot-price` option.

Auto Scaling will request Spot Instances using the maximum price, but this price is not what you actually pay for the instances when they are launched. You pay the current Spot Price as long as it is lower than your bid price. For example, say you bid \$0.05 for m1.small instances. Your bid gets fulfilled if the current Spot market price for an m1.small Spot Instance is below \$0.05. If the current price is \$0.03, then you will be charged the current market price of \$0.03 per hour. In fact, as long as your Spot bid is higher than the Spot market price, your bid will be fulfilled and a Spot Instance will be launched for you. If the current Spot market price drops to \$0.02 or rises to \$0.04, you pay the new market price.

It is important that you carefully consider the price you specify for your bid. Only specify Spot bid prices that you are willing to pay. If you intentionally bid too high, you might end up paying your high bid price if the Spot market price rises. Also, if you specify a new Spot bid price that is higher than the bid price for Spot Instances that you are currently running, your new Spot bid may result in your own running instances being replaced at the higher price. To help you choose the appropriate price to bid, you can view the Spot price history using the AWS Management Console, CLI, or API. For more information, see [View Spot Price History](#) in the *Amazon EC2 User Guide for Linux Instances*.

In addition, if you want to specify a new Spot bid price, you have to create a new launch configuration, and update your Auto Scaling group to specify the new launch configuration. Launch configurations represent a record of the details of running and terminated instances. They can be helpful in tracking the history of your instances.

For more information about changing Spot bids, see [Update the Bid Price for the Spot Instances \(p. 179\)](#) later in this topic.

For this procedure, specify the following values for the `as-create-launch-config` command:

- Launch configuration name = `spot1c-5cents`

Note

When Auto Scaling launches instances, it does not distinguish the Spot Instances from the On-Demand instances. To help you identify which of your instances are Spot Instances, refer to their launch configurations. Consider assigning a launch configuration name that includes *spot* and the bid price.

- Image ID = `ami-e565ba8c`
If you don't have an AMI, and you want to find a suitable one, follow the instructions in [Finding a Suitable AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.
- Instance type = `m1.small`
- Spot price = `$0.05`

Your command should look similar to the following example:

```
as-create-launch-config spotlc-5cents --image-id ami-e565ba8c --instance-type m1.small --spot-price "0.05"
```

You should get a confirmation similar to the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Create your Auto Scaling group by using `as-create-auto-scaling-group` and then specifying the launch configuration you just created. In addition, let Auto Scaling know at what level it should maintain your fleet of Spot Instances by setting `min-size` and `desired-capacity`.

Note

The desired capacity option tells Auto Scaling the number of instances you want the service to maintain. When you use the Auto Scaling console, you set desired capacity by specifying a value for **Start with**. When you use the Auto Scaling command line interface (CLI), you set desired capacity by specifying the `desired-capacity` option in the `update-autoscaling-group` command.

If your Spot bid price falls below the Spot Price, and Amazon EC2 terminates your instance, Auto Scaling submits your bid again and launch an instance in an effort to maintain the desired capacity you specified. However, whether Auto Scaling successfully launches an instance depends on the Spot bid price as compared to the Spot market price: If the Spot bid price is higher than the Spot market price, then an instance is launched. If the market price is higher than the bid price, then no instance is launched at that point, but Auto Scaling keeps on trying.

For more information about the syntax of the `as-create-auto-scaling-group` command, see [Create an Auto Scaling Group \(p. 36\)](#).

Specify these values for the command:

- Auto Scaling Group name = `spotasg`
- Launch configuration name = `spotlc-5cents`
- Availability Zone = `us-east-1a,us-east-1b`
- Max size = 5
- Min size = 1
- Desired capacity = 3

Your command should look similar to the following example:

```
as-create-auto-scaling-group spotasg --launch-configuration spotlc-5cents --availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 1 --desired-capacity 3
```

You should get a confirmation that looks similar to the following example:

```
OK-Created AutoScalingGroup
```

Verify and Check Your Instances

You might want to verify details of your Auto Scaling group after you've set it up.

- Check whether Auto Scaling is submitting your bids successfully.

The `as-describe-scaling-activities` command is a useful tool for this task. The command lists the activities that Auto Scaling performed for a specified Auto Scaling group.

This is the basic syntax:

```
as-describe-scaling-activities [ActivityIds [ActivityIds ...]] [--auto-scaling-group value] [--max-records value] [General Options]
```

Specifying the Auto Scaling group and the activity ID is optional. Activity ID is an identifier for an Auto Scaling activity.

If you don't specify the Auto Scaling group, the command returns all activities for all Auto Scaling groups associated with your account. If you specify the Auto Scaling group, only the activities for that Auto Scaling group will be listed.

In this procedure, you use the `as-describe-scaling-activities` command to see the state of your bid. Your command should look similar to the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --headers
```

If not all your bids are fulfilled, you get information that looks similar to the following example:

ACTIVITY NAME	ACTIVITY-ID CODE	MESSAGE	END-TIME	GROUP-
ACTIVITY	31bcbb67-7f50-4b88-ae7e-e564a8c80a90	WaitingForSpotInstanceId Placed Spot instance request: sir-fc8a3014.		spotasg
		Waiting for instance(s)		
ACTIVITY	770bbeb5-407c-404c-a826-856f65db1c57	WaitingForSpotInstanceId Placed Spot instance request: sir-69101014.		spotasg
		Waiting for instance(s)		
ACTIVITY	597e4ebd-220e-42bc-8ac9-2bae4d20b8d7	Successful	2012-05-23T17:40:22Z	spotasg

In this response, you know that your bids were placed, one of the bids is successful, and Auto Scaling is waiting for the other two bids.

Note

If the `as-describe-scaling-activities` command returns a list that includes *Failed* activities, check the data you specified in the launch configuration. For example:

- The Amazon Machine Image (AMI) might not be valid anymore.
- The Spot bid price specified in the launch configuration could be lower than the Spot market price.

If you run the `as-describe-scaling-activities` command again later, you can get information that is similar to the following example:

ACTIVITY NAME	ACTIVITY-ID CODE	END-TIME	GROUP-
ACTIVITY	90630906-b40f-41a6-967a-cd6534b2dfca	2012-06-01T02:32:15Z	spotasg
	Successful		
ACTIVITY	a1139948-ad0c-4600-9efe-9dab8ce23615	2012-06-01T00:48:02Z	spotasg
	Successful		
ACTIVITY	33001e70-6659-4494-a817-674d1b7a2f58	2012-06-01T02:31:11Z	spotasg
	Successful		

The output shows that the listed activities were successful. Because you know that *spotasg* is an Auto Scaling group that uses a launch configuration with a Spot bid price, you can assume that the activities represent the bids placed by Auto Scaling.

- Find out more information about a scaling activity.

You can get more details about the activities and instances by using the `--show-xml` option of `as-describe-scaling-activities`. Your command should look similar to the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --show-xml
```

The information you get should be similar to the following example:

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>b5a3b43e-10c6-4b61-8e41-2756db1fb8f5</NextToken>
    <Activities>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>90630906-b40f-41a6-967a-cd6534b2dfca</ActivityId>
        <StartTime>2012-06-01T00:48:21.942Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:48:21Z a difference between desired and actual
capacity changing the desired capacity, increasing the capacity from 2 to
3.</Cause>
        <Details>{}</Details>
        <Description>Launching a new EC2 instance: i-fe30d187</Description>
        <EndTime>2012-06-01T02:32:15Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>a1139948-ad0c-4600-9efe-9dab8ce23615</ActivityId>
        <StartTime>2012-06-01T00:47:51.293Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:47:51Z an instance was taken out of service
in response to a system health-check.</Cause>
        <Details>{}</Details>
        <Description>Terminating EC2 instance: i-88ce28f1</Description>
        <EndTime>2012-06-01T00:48:02Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>33001e70-6659-4494-a817-674d1b7a2f58</ActivityId>
        <StartTime>2012-06-01T00:46:19.723Z</StartTime>
```

```
<AutoScalingGroupName>spotasg</AutoScalingGroupName>
<Cause>At 2012-06-01T00:46:19Z a difference between desired and actual
capacity changing the desired capacity, increasing the capacity from 2 to
3.</Cause>
<Details>{}</Details>
<Description>Launching a new EC2 instance: i-2c30d155</Description>
<EndTime>2012-06-01T02:31:11Z</EndTime>
</member>
...
</Activities>
</DescribeScalingActivitiesResult>
<ResponseMetadata>
  <RequestId>d02af4bc-ad8f-11e1-85db-83e1968c7d8d</RequestId>
</ResponseMetadata>
</DescribeScalingActivitiesResponse>
```

The XML output shows more detail about the Spot Instance and Auto Scaling activity.

- Cause: At 2012-06-01T00:48:21Z a difference between desired and actual capacity changing the desired capacity, increasing the capacity from 2 to 3. Description: Launching a new EC2 instance: i-fe30d187

If an instance is terminated and the number of instances falls below the desired capacity, Auto Scaling launches a new instance so that the total number of your running instances rises back to the level specified for desired capacity.

- Cause: At 2012-06-01T00:47:51Z an instance was taken out of service in response to a system health-check. Description: Terminating EC2 instance: i-88ce28f1

Auto Scaling maintains the desired number of instances by monitoring the health status of the instances in the Auto Scaling group. When Auto Scaling receives notification that an instance is *unhealthy* or terminated, Auto Scaling launches another instance to take the place of the unhealthy instance. For information about how Auto Scaling monitors the health status of instances, see [Maintaining a Fixed Number of Running EC2 Instances](#) (p. 53).

Note

Auto Scaling provides the cause of instance termination that is not the result of a scaling activity. This includes instances that have been terminated because the Spot market price exceeded their bid price.

When Auto Scaling attempts to replace terminated instances resulting from the Spot market price rising above the instances' bid price, Auto Scaling places the bid specified in the current launch configuration and attempts to launch another instance to maintain the desired capacity.

- Confirm that Auto Scaling is launching your Spot Instances according to your specifications.

To confirm that Auto Scaling is launching the instances you want in the Availability Zones you specified, use `as-describe-auto-scaling-groups`. The command shows details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see [Verify Your Auto Scaling Group Creation](#) (p. 37).

This is the basic syntax:

```
as-describe-auto-scaling-groups [AutoScalingGroupNames [AutoScalingGroup-  
Names...]] [--max-records value] [General Options]
```

Specify the `--headers` general option to show the column headers, which can make the information easier to read.

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups spotasg --headers
```

The information you get should be similar to the following example.

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY
AUTO-SCALING-GROUP	spotasg	spotlc-5cents	us-east-1b,us-east-1a	1	5	3
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG	
INSTANCE	i-2c30d155	us-east-1a	InService	Healthy	spotlc-5cents	
INSTANCE	i-fe30d187	us-east-1a	InService	Healthy	spotlc-5cents	
INSTANCE	i-c630dlbf	us-east-1a	InService	Healthy	spotlc-5cents	

You can see that Auto Scaling launched 3 instances in us-east-1a, as you specified, and they are all running.

- Get details about your Spot Instances.

In addition to using `as-describe-auto-scaling-groups`, you can find out details about the Spot Instances launched for you by Auto Scaling, by using the `as-describe-auto-scaling-instances` command.

This is the basic syntax:

```
as-describe-auto-scaling-instances [InstanceIds [InstanceIds ...]] [--max-records value] [General Options]
```

Specifying `InstanceIds` is optional. If you specify it, the command returns information about the instance, if it exists. If you don't specify `InstanceIds`, the command returns information about all instances associated with your Auto Scaling account.

In this procedure, it is assumed that you created one launch configuration and Auto Scaling group, and you want to find out details about all your Spot Instances.

Your command should look similar to the following example:

```
as-describe-auto-scaling-instances --headers
```

The information you get should be similar to the following example:

INSTANCE	INSTANCE-ID	GROUP-NAME	AVAILABILITY-ZONE	STATE	STATUS
LAUNCH-CONFIG					
INSTANCE	i-2c30d155	spotasg	us-east-1a	InService	HEALTHY
spotlc-5cents					
INSTANCE	i-c630dlbf	spotasg	us-east-1a	InService	HEALTHY
spotlc-5cents					
INSTANCE	i-fe30d187	spotasg	us-east-1a	InService	HEALTHY
spotlc-5cents					

Get Notifications When the Auto Scaling Group Changes

Optionally, you can receive notifications when instances are terminated and launched. When the Spot market price rises above the Spot bid price of your running instances, Amazon EC2 terminates these instances. If your Spot Instances are terminated, Auto Scaling will try to launch replacement instances and

satisfy the desired capacity you specified for your Auto Scaling group. You can set up Auto Scaling to notify you using Amazon SNS when instance launch or termination events occur.

To do this, you will need the following:

- An Amazon Resource Name (ARN), which you generate when you create an Amazon SNS topic. An endpoint, such as an email address, must be subscribed to the topic in order for the endpoint to receive messages published to the topic. For more information, see [Create a Topic](#) in the *Amazon Simple Notification Service Developer Guide*.
- An Auto Scaling group, which you created earlier.
- A notification configuration. You configure an Auto Scaling group to send notifications when specified events take place by calling the `as-put-notification-configuration` CLI command or the `PutNotificationConfiguration` API action. For more information about the command, go to [PutNotificationConfiguration](#) in the *Auto Scaling API Reference*.
- A list of Auto Scaling notification types, which are events that cause the notification to be sent.

For information about setting up email notifications in Auto Scaling, see [Getting Notifications When Your Auto Scaling Group Changes](#) (p. 248).

Update the Bid Price for the Spot Instances

Auto Scaling launch configurations cannot be changed. They represent a record of the launch configuration details of running and terminated instances. They can be helpful in tracking the history of your instances. If you want to modify your bid price for Spot Instances, you must create a new launch configuration.

For example, if you want to launch a set of Spot Instances that have a higher likelihood of running uninterrupted for a long time, you can specify a higher Spot bid price. To do this, you must create a new launch configuration and associate it with your Auto Scaling group, using the same procedure that you followed earlier.

To update the bid price for Spot Instances

1. Create a new launch configuration specifying the new Spot bid price, by using the `as-create-launch-config` command.

Specify the following values:

- Launch configuration name = `spot1c-7cents`
- Image ID = `ami-e565ba8c`
- Instance type = `m1.small`
- Spot price = `$0.07`

Your command should look similar to the following example:

```
as-create-launch-config spot1c-7cents --image-id ami-e565ba8c --instance-type m1.small --spot-price "0.07"
```

2. Modify your Auto Scaling group to specify the new launch configuration, by using the `as-update-auto-scaling-group` command.

This is the basic syntax:

```
as-update-auto-scaling-group AutoScalingGroupName [--availability-zones value[,value...]] [--default-cooldown value] [--desired-capacity value]
```



```
[--grace-period value] [--health-check-type value] [--launch-configuration value] [--max-size value] [--min-size value] [--placement-group value] [--vpc-zone-identifier value] [General Options]
```

In this procedure, the only change to the *spotasg* Auto Scaling group is the launch configuration that it uses.

Your command, specifying the *spotasg* Auto Scaling group and the *spotlc-7cents* launch configuration, should look similar to the following example:

```
as-update-auto-scaling-group spotasg --launch-configuration spotlc-7cents
```

3. View your changes.

You can view the status of your Spot bid and a list of the bids that Auto Scaling placed for you by running *as-describe-scaling-activities* soon after you create your Auto Scaling group.

Your command should look similar to the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --headers
```

If not all your bids are fulfilled, you will get information that looks similar to the following example:

ACTIVITY NAME	ACTIVITY-ID CODE	MESSAGE	END-TIME	GROUP-
ACTIVITY	5879cc50-1e40-4539-a754-1cb084f1aecd	WaitingForSpotInstanceId Placed Spot instance request: sir-93828812. Waiting for instance(s)		spotasg
ACTIVITY	777fbelb-7a24-4aaf-b7a9-d368d0511878	WaitingForSpotInstanceId Placed Spot instance request: sir-016cf812. Waiting for instance(s)		spotasg
ACTIVITY	f4b00f81-eaea-4421-80b4-a2e3a35cc782	WaitingForSpotInstanceId Placed Spot instance request: sir-cf60ea12. Waiting for instance(s)		spotasg
ACTIVITY	31bcbb67-7f50-4b88-ae7e-e564a8c80a90	WaitingForSpotInstanceId Placed Spot instance request: sir-fc8a3014. Waiting for instance(s)		spotasg
ACTIVITY	770bbeb5-407c-404c-a826-856f65db1c57	WaitingForSpotInstanceId Placed Spot instance request: sir-69101014. Waiting for instance(s)		spotasg
ACTIVITY	597e4ebd-220e-42bc-8ac9-2bae4d20b8d7	Successful	2012-05-23T17:40:22Z	spotasg
ACTIVITY	eca158b4-a6f9-4ec5-a813-78d42c1738e2	Successful	2012-05-23T17:40:22Z	spotasg
ACTIVITY	1a5bd6c6-0b0a-4917-8cf0-eee1044a179f	Successful	2012-05-23T17:22:19Z	spotasg
ACTIVITY	c285bf16-d2c4-4ae8-acad-7450655facb5	Successful	2012-05-23T17:22:19Z	spotasg
ACTIVITY	127e3608-5911-4111-906e-31fb16d43f2e	Successful	2012-05-23T15:38:06Z	spotasg

ACTIVITY	bfb548ad-8bc7-4a78-a7db-3b41f73501fc	2012-05-23T15:38:07Z	spotasg
	Successful		
ACTIVITY	82d2b9bb-3d64-46d9-99b6-054a9ecf5ac2	2012-05-23T15:30:28Z	spotasg
	Successful		
ACTIVITY	95b7589b-f8ac-49bc-8c83-514bf664b4ee	2012-05-23T15:30:28Z	spotasg
	Successful		
ACTIVITY	57bbf77a-99d6-4d94-a6db-76b2307fb9de	2012-05-23T15:16:34Z	spotasg
	Successful		

Bids are represented by values such as `sir-93828812` and `sir-016cf812`.

When you create a new launch configuration that sets a new bid price for Spot Instances, and you have Spot Instances already running based on a different bid price, these instances based on a different bid price continue running and are only terminated if the Spot market price goes above the bid price on which the running Spot Instance was based, or if you manually terminate them.

Set Up a Schedule for Your Spot Bids

You can set up Auto Scaling to launch a certain number of instances at a specific time. This capability is useful when you want to take advantage of a window of time when prices are lower, for example, or you want to terminate Spot Instances at a specific time.

To set up a schedule, you use `as-put-scheduled-update-group-action`. This is the basic syntax:

```
as-put-scheduled-update-group-action ScheduledActionName --auto-scaling-group
value [--desired-capacity value] [--end-time value][--max-size value][--min-
size value] [--recurrence value][--start-time value][--time value][General Op-
tions]
```

In this procedure, use the following values to tell Auto Scaling to increase your fleet of instances to 20 within a five-minute period:

- Scheduled action name: `as-spotbid-schedule`
- Auto Scaling group: `spotasg`
- Start time: `2012-05-15T19:10:00Z`
- End time: `2012-05-15T19:15:00Z`
- Desired capacity: 20

Your command should look similar to the following example:

```
as-put-scheduled-update-group-action as-spotbid-schedule --auto-scaling-group
spotasg --desired-capacity 20 --start-time 2012-05-15T19:10:00Z --end-time 2012-
05-15T19:15:00Z
```

You should get a confirmation similar to the following example:

```
OK-Put Scheduled Update Group Action
```

To check your scheduled action, run `as-describe-scheduled-actions`.

You get information similar to the following example:

```
UPDATE-GROUP-ACTION spotasg as-spotbid-schedule 2012-05-15T19:10:00Z 20
```

Clean Up

After you're finished using your instances and your Auto Scaling group, it is a good practice to clean up. Run the `as-delete-auto-scaling-group` command with the optional `--force-delete` parameter. *Force delete* 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. If you don't specify the `--force-delete` parameter, then you cannot delete your Auto Scaling group until you have terminated all instances running in that Auto Scaling group.

Run the command with the following values:

- Auto Scaling group name = `spotasg`

Note

If you have more than one Auto Scaling group, you must run the command for each group that you want to delete.

- Force delete (optional parameter) = `--force-delete`

Your commands should look similar to the following example:

```
as-delete-auto-scaling-group spotasg --force-delete
```

Confirm that you want to delete the Auto Scaling group. After you confirm that you want to delete the Auto Scaling group, Auto Scaling deletes the group, as the following example shows:

```
Are you sure you want to delete this AutoScalingGroup? [Ny]  
OK-Deleted AutoScalingGroup
```

Tasks Completed

You just performed the following tasks:

- Created a launch configuration that launched Spot Instances.
- Created an Auto Scaling group.
- Obtained information about your Auto Scaling group and instances.
- Set up notifications.
- Updated the bid price.
- Scheduled spot requests.
- Cleaned up.

Following is the complete snippet used to perform these tasks. You can copy the snippet, replace the values with your own, and use the code to get started.

Note

The instance associated with the Auto Scaling group you just created is not launched immediately. If you run the snippet as a single code block, it could take a few minutes before you see the instance information.

```
as-create-launch-config spotlc-5cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.05"
as-create-auto-scaling-group spotasg --launch-configuration spotlc-5cents --
availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 1 --desired-
capacity 3
as-describe-scaling-activities --auto-scaling-group spotasg --headers
as-describe-scaling-activities --auto-scaling-group spotasg --show-xml
as-describe-auto-scaling-groups spotasg --headers
as-describe-auto-scaling-instances --headers
as-put-notification-configuration spotasg --topic-arn arn:placeholder:MyTopic
--notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN
STANCE_TERMINATE
as-describe-notification-configurations spotasg -headers
as-create-launch-config spotlc-7cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.07"
as-update-auto-scaling-group spotasg --launch-configuration spotlc-7cents
as-describe-scaling-activities --auto-scaling-group spotasg --headers
as-put-scheduled-update-group-action as-spotbid-schedule --auto-scaling-group
spotasg --desired-capacity 20 --start-time 2012-05-15T19:10:00Z --end-time 2012-
05-15T19:15:00Z
as-delete-auto-scaling-group spotasg --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 \(p. ?\)](#).

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 \(p. ?\)](#).

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 \(p. ?\)](#).

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 \(p. ?\)](#).

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 \(p. ?\)](#).

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 \(p. ?\)](#).

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 \(p. ?\)](#).

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? \(p. 1\)](#). This section describes the core concepts that you should understand before adding Auto Scaling to your network infrastructure.
- [Getting Started with Auto Scaling \(p. 28\)](#). Here, you can 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 \(p. 44\)](#). This section describes how to create launch configurations, build Auto Scaling groups, and perform other tasks associated with creating new Auto Scaling groups.

Load Balance Your Auto Scaling Group

When you use Auto Scaling, you can increase the number of Amazon Elastic Compute Cloud (Amazon EC2) instances you're using automatically 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.

This topic shows you how you can use Elastic Load Balancing to route traffic to EC2 instances in your Auto Scaling group. If you aren't already acquainted with basic Auto Scaling concepts, see [What is Auto Scaling? \(p. 1\)](#). For information about Elastic Load Balancing, see [What Is Elastic Load Balancing?](#)

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 \(p. 186\)](#).

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 \(p. 57\)](#).

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 \(p. 195\)](#).

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. 197).

Set Up a Scaled and Load-Balanced Application

The following sections step you through the process of registering your Auto Scaling group with a load balancer to set up an auto-scaled and load-balanced application.

Prerequisites

Before you begin registering your load balancer with your Auto Scaling group, be sure you have completed the following prerequisites:

- Sign up for AWS.

If you haven't yet signed up, go to <http://aws.amazon.com>, click **Sign Up**, and follow the on-screen instructions.

- Follow the steps in [Get Started With Elastic Load Balancing](#), to create a load balancer. When creating your load balancer, you can skip the step for registering your EC2 instances.

Note

You don't need to register your EC2 instances with your load balancer if you plan to attach your load balancer to an Auto Scaling group. Auto Scaling launches EC2 instances when you create your Auto Scaling group and then attach the Auto Scaling group to the load balancer.

- You can use the AWS Management Console, the Auto Scaling command line interface (CLI), or the Query API to set up a scaled and load-balanced application. For information about installing the CLI, see [Install the Auto Scaling CLI](#) (p. 16). For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs](#) (p. 23).

Topics

- [Setting up an Application Using the AWS Management Console](#) (p. 186)
- [Setting Up an Application Using the Command Line Interface](#) (p. 190)
- [Using the Query API](#) (p. 192)

Setting up an Application Using the AWS Management Console

In this section, you create a new launch configuration `my-test--lc`, create an Auto Scaling group `my-test-asg-lbs` with the desired number of instances for the Auto Scaling group, and register `my-test-asg-loadbalancer` that you created in the previous step.

Create a Launch Configuration

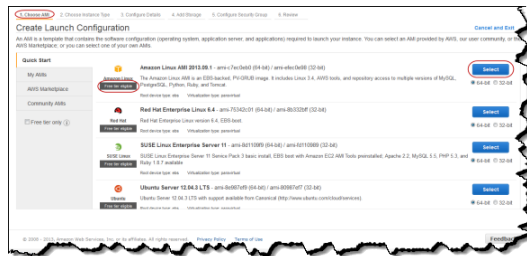
In this procedure, you either create a new launch configuration `my-test-lc` or select an existing launch configuration.

To create or select a launch configuration

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Launch Configurations**.
3. If you do not have any Auto Scaling resources in the selected region, you see **Welcome to Auto Scaling** page. Click **Create Auto Scaling group** and then click **Create launch configuration**.

If you already have Auto Scaling resources in the selected region, you see your launch configurations listed. If you want to create a new launch configuration for this procedure, click **Create launch configuration**. If you want to use an existing launch configuration, select your launch configuration, click **Create Auto Scaling group**, and then skip the following steps.

4. On the **Create Launch Configuration** wizard, the **Choose AMI** page displays a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instance. Select an AMI for your instance. This procedure uses the 64-bit Amazon Linux AMI.



5. On the **Choose Instance Type** page, you can select the hardware configuration of your instance. This procedure uses the `t1.micro` instance that is selected by default. Click **Next: Configure details** to let the **Create Launch Configuration** wizard complete other configuration settings for you, so that you can get started quickly.
6. On the **3. Configure Details** page, in the **Name** field, enter a name of your launch configuration (*my-first-1c*). Leave the other fields blank.
7. This procedure uses the storage device that comes with the AMI and assumes that you have already created a security group before you started this tutorial. Click **Skip to Review**.

To add additional storage devices and configure a security group, click **Next: Add Storage** and follow the instructions on the **Add Storage** and **Configure Security Group** pages.

8. On the **Review** page, review the details of your launch configuration.

Note

If you have not already selected a security group, the Launch Configuration wizard automatically defines the `AutoScaling-Security-Group-x` security group to allow you to connect to your instance. The `AutoScaling-Security-Group-x` security group enables all IP addresses (0.0.0.0/0) to access your instance over the specified ports.

If you would like to use a different security group, click **Edit security groups** on the bottom right, follow the instructions on the **Configure Security Group** page to choose an existing security group or create a new one. Click **Review** to continue reviewing your launch configuration.

Similarly, if you want to change any other details, click **Edit** on the right side of the field yto change.

The screenshot shows the 'Create Launch Configuration' page in the AWS Management Console. The page is divided into several sections: 'AMI Details', 'Instance Type', 'Launch configuration details', and 'Storage'. The 'Instance Type' section contains a table with columns: Instance Type, ECUs, vCPUs, Memory GiB, Instance Storage (GiB), EBS-Optimized Available, and Network Performance. The 't1.micro' instance is selected. The 'Launch configuration details' section shows various settings like Name, Purchasing option, EBS Optimized, Monitoring, IAM role, Tenancy, Keyset ID, IAM Disk ID, and User data. The 'Storage' section is also visible at the bottom.

Instance Type	ECUs	vCPUs	Memory GiB	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t1.micro	up to 2	1	0.613	EBS only	-	Very Low

9. After you are done reviewing your launch configuration, 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.
11. Select the acknowledgement check box and then click **Create Launch Configuration** to create your launch configuration.
12. 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

To create an Auto Scaling group

1. On the **Configure Auto Scaling Group Details** page, enter the following details:
 - a. In the **Group name** field, enter a name for your Auto Scaling group *my-test-asg-lbs*.
 - b. In the **Group size** field, enter *2* for the number of instances you want your Auto Scaling group to start with.
 - c. Leave the **Network** field blank for this procedure.
 - d. Click the **Availability Zone(s)** field, and select your Availability Zone *us-west-2a*.
2. Click **Advanced Details**.
3. In the **Load Balancing** field, select *Receive traffic from Elastic Load Balancer(s)*.
4. Click the empty field and select *my-test-asg-loadbalancer*.
5. In the **Health Check Type** field, select **ELB** if you want the Elastic Load Balancing health check for your Auto Scaling group.
6. The **Health Check Grace Period** field is pre-populated with the default value. You can type in the field to change the default value.

Note

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 match the expected startup period of your application.

7. Click **Next: Configure scaling policies**.
8. In the **Configure scaling policies** page, select **Keep this group at its initial size** for this procedure.

If you want to configure scaling policies for your Auto Scaling group, see [Scaling Based on Metrics \(p. 60\)](#) and follow the instructions for creating scaling policies and CloudWatch alarms using the console.
9. Click **Review** to verify the details of your Auto Scaling group. You can click **Edit** to change the details of your Auto Scaling group.
10. Click **Create Auto Scaling Group**
11. The **Auto Scaling Group creation status** page lets you know that your Auto Scaling group was successfully created.

Verify That Your Auto Scaling Group Launched with our Load Balancer

In this section, you use the Amazon EC2 console to verify that your Auto Scaling group has launched with your load balancer.

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

1. In the **Auto Scaling Group creation status** page, click **Close**.
2. In the **Auto Scaling Groups** page, select *my-test-asg-lbs*.
3. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab.

The **Load Balancers** field in the **Details** tab displays *my-test-asg-loadbalancer*.
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.
6. 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 2 new instances, and it is in *InService* state. The **Health Status** column shows the result of the health check on your instances.

Setting Up an Application Using the Command Line Interface

In this section, you create a launch configuration `my-test-lc`, an Auto Scaling group `my-test-asg-lbs`, and register the load balancer `my-test-asg-loadbalancer` that you created in the previous step.

Create a Launch Configuration

If you'd rather use your own launch configuration, skip the following procedure.

To create the launch configuration

1. Use the `as-create-launch-config` command and specify the following values:

- Launch configuration name = `my-test-lc`
- Image ID = `ami-514ac838`

Note

The image ID is provided for illustration purposes only. Image IDs change over time. To obtain a list of current valid image IDs, see [Finding a Suitable AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

- Instance type = `m1.small`

Your command should look similar to the following example:

```
as-create-launch-config my-test-lc --image-id ami-514ac838 --instance-type m1.small
```

2. You should get a confirmation similar to the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

You can use the following procedure to create an Auto Scaling group and attach a load balancer.

To create an Auto Scaling group

1. Use the `as-create-auto-scaling-group` command and specify the following values:

- Auto Scaling group name = `my-test-asg-lbs`
- Launch configuration name = `my-test-lc`
- Availability Zone = `us-east-1a,us-east-1b`
- Load Balancer name = `my-test-asg-loadbalancer`
- Max size = 5
- Min size = 1
- Desired capacity = 2

Note

The load of the incoming traffic is balanced equally across all Availability Zones enabled for your load balancer. Auto Scaling tries to launch an equivalent number of instances in each

zone. As a best practice, we recommend that you specify even numbers for maximum, minimum, and desired capacity for your Auto Scaling group.

Your command should look similar to the following example:

```
as-create-auto-scaling-group my-test-asg-lbs --launch-configuration my-test-
lc --availability-zones
us-east-1a,us-east-1b --load-balancers my-test-asg-loadbalancer --max-size
5 --min-size 1 --desired-capacity 2
```

2. You should get a confirmation similar to the following example:

```
OK-Created AutoScalingGroup
```

Verify That Your Auto Scaling Group Launched with a Load Balancer

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

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

1. Use the `as-describe-auto-scaling-groups` command and specify the following value:
 - Auto Scaling group name = `my-test-asg-lbs`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg-lbs --headers
```

Note

Specify the `--headers` general option to show column headers that will organize the describe command information.

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

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	LOAD-BALANCERS	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg-lbs	my-test-lc	us-east-1b,us-east-1a	my-test-asg-loadbalancer	1	5	2	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG			
INSTANCE	i-78e60b1b	us-east-1b	InService	Healthy	my-test-lc1			
INSTANCE	i-941599fe	us-east-1a	InService	Healthy	my-test-lc1			

You can see that Auto Scaling has registered the load balancer `my-test-asg-loadbalancer` and launched two instances using the launch configuration `my-test-lc`.

Using the Query API

In this walkthrough, you'll create a launch configuration `my-test-lc`, an Auto Scaling group `my-test-asg-lbs`, and register the load balancer `my-test-asg-loadbalancer` created in the previous step, with the Auto Scaling group.

Create Launch Configuration

If you'd rather use your own launch configuration, skip the following procedure.

To create the launch configuration

1. Call the [CreateLaunchConfiguration](#) action and specify the following parameters:

- `LaunchConfigurationName` = `my-test-lc`
- `ImageId` = `ami-514ac838`

Note

The image ID is provided for illustration purposes only. Image IDs change over time. To obtain a list of current valid image IDs, see [Finding a Suitable AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

- `InstanceType` = `m1.small`

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?LaunchConfigurationName=my-test-lc
&ImageId=ami-514ac838
&InstanceType=m1.small
&Action=CreateLaunchConfiguration
&AUTHPARAMS
```

2. If your request was successful, you should get a confirmation similar to the following example:

```
<CreateLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7c6e177f-f082-11e1-ac58-3714bEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateLaunchConfigurationResponse>
```

Create a Auto Scaling Group

The following procedure walks you through the process of creating an Auto Scaling group and attaching a load balancer.

To create an Auto Scaling group

1. Call the [CreateAutoScalingGroup](#) action and specify the following parameters:

- `AutoScalingGroupName` = `my-test-asg-lbs`
- `LaunchConfigurationName` = `my-test-lc`
- `AvailabilityZones.member.1` = `us-east-1a`
- `AvailabilityZones.member.2` = `us-east-1b`

- LoadBalancerNames.member.1 = *my-test-asg-loadbalancer*
- MaxSize = 5
- MinSize = 1
- DesiredCapacity = 2

Note

The load of the incoming traffic is balanced equally across all Availability Zones enabled for your load balancer. Auto Scaling tries to launch an equivalent number of instances in each zone. As a best practice, we recommend that you specify even numbers for maximum, minimum, and desired capacity for your Auto Scaling group.

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg-lbs
&LoadBalancerNames.member.1=my-test-asg-loadbalancer
&AvailabilityZones.member.1=us-east-1a
&AvailabilityZones.member.2=us-east-1b
&MinSize=1
&MaxSize=5
&DesiredCapacity=2
&LaunchConfigurationName=my-test-lc
&Version=2011-01-01
&Action=CreateAutoScalingGroup
&AUTHPARAMS
```

2. If your request was successful, you should get a confirmation similar to the following example:

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>00bc5d78-8eb7-11e2-95f9-c35bfEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

Verify That Your Auto Scaling Group Launched with a Load Balancer

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

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

1. Call the [DescribeAutoScalingGroups](#) action and specify the following parameter:
 - AutoScalingGroupNames.member.1 = my-test-asg-lbs

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-
asg-lbs
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

2. The response includes details about the group and instances launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg-lbs</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2012-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1b</AvailabilityZone>
            <InstanceId>i-78e60b1b</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-941599fe</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1b</member>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames>
          <member>my-test-asg-loadbalancer</member>
        </LoadBalancerNames>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:e8b084c9-8cad-444a-8381-c97b778e0fc0:auto
ScalingGroupName/my-test-asg-l
bs</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>95fdfeb8-aa76-11e2-81e1-750aaEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

You can see that Auto Scaling registered the load balancer `my-test-asg-loadbalancer` and launched two instances using the launch configuration `my-test-lc`. The newly launched instances are healthy, and running (`InService`).

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*.

This topic shows 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. 186).

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

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.

You can add an Elastic Load Balancing health check to your Auto Scaling group using the AWS Management Console, the Auto Scaling command line interface (CLI), or the Query API. For information about installing the CLI, see [Install the Auto Scaling CLI](#) (p. 16). For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs](#) (p. 23).

Topics

- [Adding a Health Check Using the Console](#) (p. 195)
- [Adding a Health Check Using the Command Line Interface](#) (p. 196)
- [Adding a Health Check Using the Query API](#) (p. 196)

Adding a Health Check Using the Console

To add an Elastic Load Balancing health check using the console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the **Auto Scaling Groups** page, select `my-test-asg-lbs`.
4. The bottom pane displays the details of your Auto Scaling group. elect the **Details** tab
5. Click **Edit**.
6. Click the **Health Check Type** field and select `ELB`.

7. In the **Health Check Grace Period** field, enter 300.

Note

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 match the expected startup period of your application.

8. Click **Save**.
9. In the bottom pane, click the **Instances** tab.
10. 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.

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

Adding a Health Check Using the Command Line Interface

To add an Elastic Load Balancing health check, use the `as-update-auto-scaling-group` command and specify the following values:

- Auto Scaling group name = `my-test-asg-lbs`
- Health check type = `ELB`
- Health check grace period = `300`.

Note

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 match the expected startup period of your application.

Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg-lbs --health-check-type ELB --grace-period 300
```

You should get a confirmation similar to the following example:

```
OK-Updated AutoScalingGroup
```

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

Adding a Health Check Using the Query API

To add an Elastic Load Balancing health check, call the `UpdateAutoScalingGroup` action by specifying the following parameters:

- `AutoScalingGroupName` = `my-test-asg-lbs`
- `HealthCheckType` = `ELB`
- `HealthCheckGracePeriod` = `300`

Note

Frequently, new instances need to arm 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 match the expected startup period of your application.

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?HealthCheckType=ELB
&HealthCheckGracePeriod=300
&AutoScalingGroupName=my-test-asg-lbs
&Version=2011-01-01
&Action=UpdateAutoScalingGroup
&AUTHPARAMS
```

If your request was successful, you should get a confirmation similar to the following example:

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>adafead0-ab8a-11e2-ba13-ab0ccEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

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

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 designated 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 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. For information about the regions and Availability Zones supported by Auto Scaling, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

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 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 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.

You can use the Auto Scaling command line interface along with the Elastic Load Balancing CLI to add an Availability Zone to your application. You can also use the Query API. For information about installing the Auto Scaling CLI, see [Install the Auto Scaling CLI \(p. 16\)](#). For information about installing the Elastic Load Balancing CLI, see [Installing the Command Line Interface](#) in the *Elastic Load Balancing Developer Guide*. For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs \(p. 23\)](#).

Topics

- [Expanding Applications Using the Command Line Interface \(p. 198\)](#)
- [Expanding Applications Using the Query API \(p. 199\)](#)

Expanding Applications Using the Command Line Interface

In this procedure, expand the availability of your application to an additional Availability Zone, `us-east-1c`.

To expand a scaled, load-balanced application to an additional Availability Zone using the CLI

1. Use the `as-update-auto-scaling-group` command and specify the following values:

- Auto Scaling group name = `my-test-asg-lbs`
- Availability Zones = `us-east-1a`, `us-east-1b`, `us-east-1c`
- Min size = 3

Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg-lbs --availability-zones us-east-1a, us-east-1b, us-east-1c --min-size 3
```

You should get a confirmation similar to the following example:

```
OK-Updated AutoScalingGroup
```

2. Use the `as-describe-auto-scaling-groups` command and specify the following value:

- Auto Scaling group name = `my-test-asg-lbs`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg-lbs --headers
```

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

```
AUTO-SCALING-GROUP  GROUP-NAME          LAUNCH-CONFIG  AVAILABILITY-ZONES
LOAD-BALANCERS     MIN-SIZE    MAX-SIZE    DESIRED-CAPACITY  TERMINATION-POLICIES
```

Auto Scaling Developer Guide

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

AUTO-SCALING-GROUP my-test-asg-lbs my-test-lc us-east-1c,us-east-1b,us-east-1a my-test-asg-loadbalancer 3 6 3 Default					
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG
INSTANCE	i-78e60b1b	us-east-1b	InService	Healthy	my-test-lc
INSTANCE	i-dd4b2eb2	us-east-1c	InService	Healthy	my-test-lc
INSTANCE	i-48a1cf29	us-east-1a	InService	Healthy	my-test-lc

When the status of each of the new instances in the Auto Scaling group in the new Availability Zone changes to `InService`, this indicates that the instances are now ready to accept traffic from Elastic Load Balancing. You can then proceed to the next step.

Note

When you call the `elb-enable-zones-for-lb` command, the load balancer begins to route traffic equally among all of the enabled Availability Zones.

3. Use the Elastic Load Balancing `elb-enable-zones-for-lb` command by specifying the following values:
 - Load balancer name = `my-test-asg-loadbalancer`
 - Availability Zone = `us-east-1c`

Your command should look similar to the following example:

```
elb-enable-zones-for-lb my-test-asg-loadbalancer --availability-zones
us-east-1c --headers
```

Elastic Load Balancing responds with a list of Availability Zones enabled for the load balancer. The information you get should be similar to the following example:

```
AVAILABILITY_ZONES  AVAILABILITY-ZONES
AVAILABILITY_ZONES  us-east-1a, us-east-1b, us-east-1c
```

You have load-balanced your Amazon EC2 application across three Availability Zones and have scaled it in each zone.

Expanding Applications Using the Query API

In this example, you learn how to expand the availability of your application to an additional Availability Zone, `us-east-1c`.

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

1. Call the [UpdateAutoScalingGroup](#) action and specify the following parameters:
 - `AutoScalingGroupName` = `my-test-asg`
 - `AvailabilityZones.member.1` = `us-east-1a`
 - `AvailabilityZones.member.2` = `us-east-1b`
 - `AvailabilityZones.member.3` = `us-east-1c`
 - `MinSize` = 3

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg-lbs
&AvailabilityZones.member.1=us-east-1a
&AvailabilityZones.member.2=us-east-1b
&AvailabilityZones.member.3=us-east-1c
&MinSize=3
&Version=2011-01-01
&Action=UpdateAutoScalingGroup
&AUTHPARAMS
```

If your request was successful, you should get a confirmation similar to the following example:

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>325f71b3-ac46-11e2-9cae-61a8EXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

2. Call the [DescribeAutoScalingGroups](#) action and specify the following parameter:

- AutoScalingGroupNames.member.1 = my-test-asg-lbs

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-
asg-lbs
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

3. The response includes details about the group and instances launched. The information you get should be similar to the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg-lbs</AutoScalingGroupName>
        <HealthCheckType>ELB</HealthCheckType>
        <CreatedTime>2013-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-44a7b627</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
```

```
<HealthStatus>Healthy</HealthStatus>
<AvailabilityZone>us-east-1b</AvailabilityZone>
<InstanceId>i-c34f20a3</InstanceId>
<LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
<LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>Healthy</HealthStatus>
  <AvailabilityZone>us-east-1c</AvailabilityZone>
  <InstanceId>i-98562cf1</InstanceId>
  <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
</Instances>
<DesiredCapacity>3</DesiredCapacity>
<AvailabilityZones>
  <member>us-east-1c</member>
  <member>us-east-1b</member>
  <member>us-east-1a</member>
</AvailabilityZones>
<LoadBalancerNames>
  <member>my-test-asg-loadbalancer</member>
</LoadBalancerNames>
<MinSize>3</MinSize>
<VPCZoneIdentifier/>
<HealthCheckGracePeriod>300</HealthCheckGracePeriod>
<DefaultCooldown>300</DefaultCooldown>
<AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:e8b084c9-8cad-444a-8381-c97b778e0fc0:auto
ScalingGroupName/my-test-asg-lbs</AutoScalingGroupARN>
<TerminationPolicies>
  <member>Default</member>
</TerminationPolicies>
<MaxSize>6</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>084332c2-ac57-11e2-b92b-45061efc08bd</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Check the status of the instances in the `<LifecycleState>` field of all three instances. When the status of each of the new instances in the new Availability Zone appears as `InService`, indicating that the instances are now recognized by the load balancer and ready, proceed to the next step.

Note

When you call `EnableAvailabilityZonesForLoadBalancer`, the load balancer begins to route traffic equally among all the enabled Availability Zones.

4. Call the [EnableAvailabilityZonesForLoadBalancer](#) action by specifying the following parameters:
 - `AvailabilityZones` = `us-east-1c`
 - `LoadBalancerName` = `my-test-asg-loadbalancer`

Your request should look similar to the following example:

```
https://elasticloadbalancing.amazonaws.com/?AvailabilityZones.member.1=us-east-1c
&LoadBalancerName=my-test-asg-loadbalancer
&Version=2012-06-01
&Action=EnableAvailabilityZonesForLoadBalancer
&AUTHPARAMS
```

5. The response includes a list of Availability Zones enabled for the load balancer. The information you get should be similar to the following example:

```
<EnableAvailabilityZonesForLoadBalancerResponse xmlns="http://elasticloadbalancng.amazonaws.com/doc/2012-06-01/">
  <EnableAvailabilityZonesForLoadBalancerResult>
    <AvailabilityZones>
      <member>us-east-1c</member>
      <member>us-east-1b</member>
      <member>us-east-1a</member>
    </AvailabilityZones>
  </EnableAvailabilityZonesForLoadBalancerResult>
  <ResponseMetadata>
    <RequestId>1aelf97a-ac59-11e2-ac73-fffdEXAMPLE</RequestId>
  </ResponseMetadata>
</EnableAvailabilityZonesForLoadBalancerResponse>
```

You have load-balanced your Amazon EC2 application across three Availability Zones and have scaled it in each zone.

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. This section walks you through the process of attaching an instance to an Auto Scaling group. Before you get started, be sure you have done the following:

- Created a launch configuration and an Auto Scaling group. If you are not familiar with how to create a launch configuration or an Auto Scaling group, follow the steps in the [Getting Started with Auto Scaling Using the CLI \(p. 34\)](#).
- Identified an EC2 instance to attach to the Auto Scaling group. You can identify the instance by looking at the descriptions of all the EC2 instances associated with your AWS account. Use the Amazon EC2 console in the AWS Management Console, the `ec2-describe-instances` Amazon EC2 command, or the `DescribeInstances` Amazon EC2 action to get the descriptions of all the instances associated with your AWS account. After you've identified the instance, make a note of the instance ID, and make sure to verify the following:
 - The identified instance is in the `running` state.
 - The identified instance is in the same Availability Zone as the Auto Scaling group.
 - The identified instance and the Auto Scaling group belong to the same AWS account.
 - The identified instance is not a member of another Auto Scaling group.
- If the Auto Scaling group is associated with a load balancer, the identified instance and the load balancer are either in EC2-Classic or in the same VPC.

- The `Launch` process is not suspended for the Auto Scaling group.

Note

Auto Scaling does not validate any other instance attributes other than those listed above. For example, if the instance has attributes such as `subnetId`, `placement` and so on, those attributes are not validated.

For this procedure, use an Auto Scaling group with the following configuration:

- Auto Scaling group name = `my-test-asg`
- Minimum size = 1
- Maximum size = 5
- Desired capacity = 2
- Availability Zone = `us-east-1a`

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.

You can attach an instance to your Auto Scaling group using the Auto Scaling CLI or the Query API. This Auto Scaling feature is currently not supported by the Amazon EC2 console.

The following sections walk you through the process of attaching instance `i-a8e09d9c` to Auto Scaling group `my-test-asg`.

Attaching Instances Using the Command Line Interface

Before attaching the instance, verify that the Auto Scaling group exists and check the number of running instances within the Auto Scaling group.

To attach an instance to an existing Auto Scaling group using the CLI

1. Enter the `as-describe-auto-scaling-groups` command with the following parameter
 - Auto Scaling group name = `my-test-asg`

Describe Auto Scaling group to get the status of the group:

```
as-describe-auto-scaling-groups my-test-asg --headers
```

Auto Scaling responds as in the following example:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1a	1	5	2	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-a5e87793	us-east-1a	InService	Healthy	my-test-lc		
INSTANCE	i-a4e87792	us-east-1a	InService	Healthy	my-test-lc		

The desired capacity of the Auto Scaling group is set to 2 and the group has 2 instances running.

2. Enter the `as-attach-instances` command with the following parameters:

- Instance ID = `i-a8e09d9c`
- Auto Scaling group name = `my-test-asg`

Attach an instance to the Auto Scaling group:

```
as-attach-instances i-a8e09d9c --auto-scaling-group my-test-asg
```

When the instance is being attached, Auto Scaling returns the following message:

```
OK-Instance(s) will be attached
```

3. Enter the `as-describe-auto-scaling-groups` command with the following parameter to verify that the instance is attached:

- Auto Scaling group name = `my-test-asg`

Describe Auto Scaling group to verify the instance is attached

```
as-describe-auto-scaling-groups my-test-asg --headers
```

The following description describes the Auto Scaling group:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1a	1	5	3	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-a8e09d9c	us-east-1a	InService	Healthy			
INSTANCE	i-a4e87792	us-east-1a	InService	Healthy	my-test-lc		
INSTANCE	i-a5e87793	us-east-1a	InService	Healthy	my-test-lc		

The desired capacity has increased by 1 and is now set to 3. The group has a new instance `i-a8e09d9c`.

Attaching Instances Using the Query API

Before attaching the instance, verify that the Auto Scaling group exists and check the number of instances running within the Auto Scaling group.

To attach an instance to an existing Auto Scaling group using the Query API

1. Use `DescribeAutoScalingGroups` action with the following parameter

- `AutoScalingGroupName` = `my-test-asg`

Describe the Auto Scaling group to get the status of the group:

```
http://autoscaling.us-east-1a.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-asg
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The following response describes the status of the Auto Scaling group:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-12-04T18:55:16.249Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <DesiredCapacity>2</DesiredCapacity>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-a5e87793</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-a4e87792</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <AvailabilityZones>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1a:605053316265:autoScalingGroup:5ca32c19-be23-4139-b000-d5dc5302c311:autoScalingGroupName/my-test-asg</AutoScalingGroupARN>
        <MaxSize>5</MaxSize>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
```

```
<ResponseMetadata>
  <RequestId>7c4cf550-5d28-11e3-802c-238155EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

The desired capacity of the Auto Scaling group is set to 2 and the group has 2 instances running.

2. Use the [AttachInstances](#) action with the following parameters:

- InstanceID = *i-a8e09d9c*
- AutoScalingGroupName = *my-test-asg*

Attach an instance:

```
http://autoscaling.us-east-1a.amazonaws.com/?InstanceIds.member.1=i-a8e09d9c
&AutoScalingGroupName=my-test-asg
&Version=2011-01-01
&Action=AttachInstances
&AUTHPARAMS
```

After attaching the instance, Auto Scaling responds as in the following example:

```
<AttachInstancesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>1be2d1de-5d3b-11e3-8e1c-a924dfEXAMPLE</RequestId>
  </ResponseMetadata>
</AttachInstancesResponse>
```

3. Use the [DescribeAutoScalingGroups](#) action with the following parameters to verify that the instance is attached to the Auto Scaling group:

- AutoScalingGroupName = *my-test-asg*

Describe the Auto Scaling group to verify that the instance is attached:

```
http://autoscaling.us-east-1a.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-asg
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The following response describes the Auto Scaling group with the attached instance:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
</DescribeAutoScalingGroupsResponse>
```

```
<HealthCheckType>EC2</HealthCheckType>
<CreatedTime>2013-12-04T18:55:16.249Z</CreatedTime>
<EnabledMetrics/>
<LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
<DesiredCapacity>3</DesiredCapacity>
<Instances>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1a</AvailabilityZone>
    <InstanceId>i-a8e09d9c</InstanceId>
    <LifecycleState>InService</LifecycleState>
  </member>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1a</AvailabilityZone>
    <InstanceId>i-a4e87792</InstanceId>
    <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
    <LifecycleState>InService</LifecycleState>
  </member>
  <member>
    <HealthStatus>Healthy</HealthStatus>
    <AvailabilityZone>us-east-1a</AvailabilityZone>
    <InstanceId>i-a5e87793</InstanceId>
    <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
    <LifecycleState>InService</LifecycleState>
  </member>
</Instances>
<AvailabilityZones>
  <member>us-east-1a</member>
</AvailabilityZones>
<LoadBalancerNames/>
<MinSize>1</MinSize>
<VPCZoneIdentifier/>
<HealthCheckGracePeriod>0</HealthCheckGracePeriod>
<DefaultCooldown>300</DefaultCooldown>
<AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:605053316265:autoScalingGroup:5ca32c19-be23-4139-b000-d5dc5302c311:autoScalingGroupName/my-test-asg</AutoScalingGroupARN>
<TerminationPolicies>
  <member>Default</member>
</TerminationPolicies>
<MaxSize>5</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>e8057206-5d29-11e3-b68e-eb6eEXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

The desired capacity has increased by 1 and is now set to 3. The group has a new instance i-a8e09d9c.

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 an Auto Scaling group.

You can easily detach an instance from one Auto Scaling group and, by following the steps described in [Attach EC2 Instances to Your Auto Scaling Group \(p. 202\)](#), attach the instance to a different Auto Scaling group.

- Test an Auto Scaling group.

You can create an Auto Scaling group using existing instances that are already a part of your application's architecture. Then, you can detach the instances from the Auto Scaling group when your tests are complete.

For this walkthrough, 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`

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.

You can detach an instance from your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. This Auto Scaling feature is currently not supported by the Amazon EC2 console.

The following sections walk you through the process of detaching instance `i-a5e87793` from Auto Scaling group `my-asg`.

Detaching Instances Using the Command Line Interface

Before detaching the instance, verify that the Auto Scaling group exists and check the number of running instances within the Auto Scaling group.

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

1. Enter the `as-describe-auto-scaling-instances` command

```
as-describe-auto-scaling-instances
```

Auto Scaling responds as in the following example:

```
INSTANCE    i-2a2d8978    my-asg    us-east-1a    InService    HEALTHY    my-asg
INSTANCE    i-5f2e8a0d    my-asg    us-east-1a    InService    HEALTHY    my-asg
INSTANCE    i-a52387f7    my-asg    us-east-1a    InService    HEALTHY    my-asg
INSTANCE    i-f42d89a6    my-asg    us-east-1a    InService    HEALTHY    my-asg
```

The desired capacity of the Auto Scaling group is set to 4 and the group has 4 instances running.

2. Enter the `as-detach-instances` command with the following parameters:

- Instance ID = `i-2a2d8978`
- Auto Scaling group name = `my-asg`
- Decrement desired capacity = `decrement-desired-capacity`

Detach the instance from the Auto Scaling group:

```
as-detach-instances i-2a2d8978 --auto-scaling-group my-asg --decrement-de
sired-capacity
```

When the instance is being detached, Auto Scaling returns a message similar to the following:

```
INSTANCE    10c2afe7-cdad-4708-b6ab-d9f3a6807425    InProgress    At 2014-06-
13T23:56:47Z instance i-2a2d8978 was detached in response to a user request,
shrinking
the capacity from 4 to 3.
```

3. Enter the `as-describe-auto-scaling-instances` command to verify that the instance is detached:

```
as-describe-auto-scaling-instances
```

The following description describes the Auto Scaling group:

```
INSTANCE    i-5f2e8a0d    my-asg    us-east-1a    InService    HEALTHY    my-asg
INSTANCE    i-a52387f7    my-asg    us-east-1a    InService    HEALTHY    my-asg
INSTANCE    i-f42d89a6    my-asg    us-east-1a    InService    HEALTHY    my-asg
```

The desired capacity has decreased by 1 and is now set to 3.

Detaching Instances Using the Query API

Before detaching the instance, verify that the Auto Scaling group exists and check the number of running instances within the Auto Scaling group.

To detach an instance from an existing Auto Scaling group using the Query API

1. Enter the `as-describe-auto-scaling-instances` command:

```
http://fws-cs1s-ga.amazon.com/?MaxRecords=20&Version=2011-01-01&Action=DescribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-14T00%3A04%3A49.251Z&AUTHPARAMS
```

Auto Scaling responds as in the following example:

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/"
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5f2e8a0d</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-a52387f7</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-e43e9ab6</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-f42d89a6</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
  <ResponseMetadata>
    <RequestId>8080b6a9-f357-11e3-bc51-b35178f0274f</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

The desired capacity of the Auto Scaling group is set to 4 and the group has 4 instances running.

2. Detach the instance

```
http://fws-cs1s-ga.amazon.com/?AutoScalingGroupName=my-asg&ShouldDecrement
```

```
DesiredCapacity=true&InstanceIds.member.1=i-5f2e8a0d&Version=2011-01-01
&Action=DetachInstances&SignatureVersion=2&SignatureMethod=Hmac
SHA256&Timestamp=2014-06-14T00%3A07%3A29.962Z&AUTHPARAMS
```

When the instance is being detached, Auto Scaling returns a message similar to the following:

```
<DetachInstancesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DetachInstancesResult>
    <Activities>
      <member>
        <ActivityId>e54ff599-bf05-4076-8b95-a0f090ed90bb</ActivityId>
        <Progress>50</Progress>
        <StatusCode>InProgress</StatusCode>
        <StartTime>2014-06-14T00:07:30.280Z</StartTime>
        <Cause>At 2014-06-14T00:07:30Z instance i-5f2e8a0d was detached in
response to a user request, shrinking the capacity from 4 to 3.</Cause>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <Details>{"Availability Zone": "us-east-1a"}</Details>
        <Description>Detaching EC2 instance: i-5f2e8a0d</Description>
      </member>
    </Activities>
  </DetachInstancesResult>
  <ResponseMetadata>
    <RequestId>e04f3b11-f357-11e3-a434-7f10009d5849</RequestId>
  </ResponseMetadata>
</DetachInstancesResponse>
```

3. Optional. Verify that the instance is detached

```
http://fws-csls-qa.amazon.com/?MaxRecords=20&Version=2011-01-01&Action=De
scribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=HmacSHA256
&Timestamp=2014-06-14T00%3A10%3A08.795Z&AUTHPARAMS
```

The following description describes the Auto Scaling group

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-a52387f7</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
</DescribeAutoScalingInstancesResponse>
```



```
<InstanceId>i-e43e9ab6</InstanceId>
<LaunchConfigurationName>my-asg</LaunchConfigurationName>
<LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-f42d89a6</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>3f1ccb13-f358-11e3-bc51-b35178f0274f</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

The desired capacity has decreased by 1 and is now set to 3.

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 Auto Scaling group, and then delete the other Auto Scaling groups.

This section walks you through the process of merging separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones.

The following examples assume that you have two identical groups: `myGroupA` in Availability Zone `us-east-1a`, and `myGroupB` in Availability Zone `us-east-1c`. The two Auto Scaling groups have the following specifications:

- Minimum size = 2
- Maximum size = 5
- Desired Capacity = 5

Note

This example works for groups with or without a load balancer, provided that the new multi-zone group will be in one of the same zones as the original single-zone groups.

You can merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones using the Auto Scaling command line interface (CLI) or the Query API.

Topics

- [Merge Zones Using the Command Line Interface \(p. 213\)](#)
- [Merge Zones Using the Query API \(p. 215\)](#)

Merge Zones Using the Command Line Interface

For information about installing the CLI, see [Install the Auto Scaling CLI \(p. 16\)](#).

To merge separate single-zone Auto Scaling groups into a single multi-zone group:

1. Use the `as-update-auto-scaling-group` command and specify the following values:

- Auto Scaling group name = `myGroupA`
- Availability Zones = `us-east-1a, us-east-1c`
- Min size = 4
- Max size = 10

Your command should look similar to the following example:

```
PROMPT>as-update-auto-scaling-group myGroupA --availability-zones us-east-1a, us-east-1c --max-size 10 --min-size 4
```

You should get a confirmation similar to the following example:

```
OK-AutoScaling Group updated
```

2. Use the `as-set-desired-capacity` command to increase the capacity of `myGroupA` to six. Specify the following values:

- Auto Scaling group name = `myGroupA`
- Desired capacity = 6

Your command should look similar to the following example:

```
PROMPT>as-set-desired-capacity myGroupA --desired-capacity 6
```

3. Use the `as-describe-auto-scaling-groups` command to check that `myGroupA` has been successfully added to the additional zones and is at the required capacity. Specify the following values:

- Auto Scaling group name = `myGroupA`

Your command should look similar to the following example:

```
PROMPT>as-describe-auto-scaling-groups myGroupA
```

4. After you've confirmed that your Auto Scaling group `myGroupA` has expanded to an additional Availability Zone `us-east-1c`, you can delete the `myGroupB` Auto Scaling group.

- a. To delete an Auto Scaling group

Use the `as-update-auto-scaling-group` command by specifying the following values:

- Auto Scaling group name = `myGroupB`
- Min size = 0
- Max size = 0

Your command should look similar to the following example:

```
PROMPT>as-update-auto-scaling-group myGroupB --min-size 0 --max-size 0
```

You should get a confirmation similar to the following example:

```
OK-AutoScaling Group updated
```

- b. Use the `as-describe-auto-scaling-groups` command to verify that no instances remain in your Auto Scaling group. Specify the following values:

- Auto Scaling group name = `myGroupB`

Your command should look similar to the following example:

```
PROMPT>as-describe-auto-scaling-groups myGroupB --headers
```

Auto Scaling returns the following:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAIL
ABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY
AUTO-SCALING-GROUP	myGroupB	MyLC	us-east-
1c	0	0	0

- c. Use the `as-describe-scaling-activities` command to verify that your scaling activities are successful. Specify the following value:

- Auto Scaling group name = `myGroupB`

Your command should look similar to the following example:

```
PROMPT>as-describe-scaling-activities myGroupB
```

Auto Scaling returns the following:

ACTIVITY	ACTIVITY-ID	END-TIME	CODE	CAUSE
ACTIVITY	74758a33-bfd5-4df...	2009-05-11T16:27:36Z	Successful	"At 2009-05-21 10:00:00Z an instance was shutdown."
ACTIVITY	74958a77-bfd5-4df...	2009-05-11T16:27:36Z	Successful	"At 2009-05-21 10:00:00Z an instance was shutdown."

- d. Use the `as-delete-auto-scaling-group` command and specify the following value:

- Auto Scaling group name = `myGroupB`

Your command should look similar to the following example:

```
PROMPT>as-delete-auto-scaling-group myGroupB
```

Auto Scaling returns the following:

```
Are you sure you want to delete this AutoScalingGroup? [Ny] y
```

- Enter `y` to delete the Auto Scaling group.

Auto Scaling returns the following:

```
OK-Deleted AutoScalingGroup
```

Merge Zones Using the Query API

You can merge your separate single-zone Auto Scaling groups into a single multi-zone group using the Query API. For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs](#) (p. 23).

To merge separate single-zone Auto Scaling groups into a single multi-zone group

1. Call the [UpdateAutoScalingGroup](#) action with the following parameters to add an additional Availability Zone to the `myGroupA` Auto Scaling group:
 - `AvailabilityZones.member.1` = `us-east-1a`
 - `AvailabilityZones.member.2` = `us-east-1c`
 - `AutoScalingGroupName` = `myGroupA`
 - `MinSize` = 4
 - `MaxSize` = 10
2. Call the [SetDesiredCapacity](#) action, and specify the following parameters to increase the capacity of `myGroupA` to six:
 - `AutoScalingGroupName` = `myGroupA`
 - `DesiredCapacity` = 6
3. Call the [DescribeAutoScalingGroups](#) action, and specify the following parameter to verify an additional zone has been successfully added to the `myGroupA` Auto Scaling group and the group is at the required capacity:
 - `AutoScalingGroupName` = `myGroupA`
4. After you've confirmed that your Auto Scaling group `myGroupA` has expanded to an additional Availability Zone `us-east-1c`, you can delete the `myGroupB` Auto Scaling group. To delete `myGroupB`:
 - a. Call [UpdateAutoScalingGroup](#) action, and specify the following parameters:
 - `AutoScalingGroupName` = `myGroupB`
 - `MinSize` = 0

- *MaxSize* = 0
- b. Verify that your changes to `myGroupB` have taken effect by doing the following:
 - i. Call the `DescribeAutoScalingGroups` action, and specify the following parameter to verify that no instances are left in the group.
 - *AutoScalingGroupName* = `myGroupB`
 - ii. Call the `DescribeScalingActivities` action, and specify the following parameter to verify that all scaling activities have completed.
 - *AutoScalingGroupName* = `myGroupB`
- c. Call the `DeleteAutoScalingGroup` action, and specify the following parameter:
 - *AutoScalingGroupName* = `myGroupB`

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.

To put an instance into standby using either the **as-enter-standby** command in the Auto Scaling CLI, the **aws autoscaling enter-standby** command in the AWS CLI, or the **EnterStandby** action. When you want to put an instance on standby, you need to know:

- The instance ID or IDs
- The Auto Scaling group name

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. This can result in Auto Scaling launching additional instances. 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.

Important

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

If you use Spot Instances as part of your Auto Scaling group, and the Spot Price increases past your bid price, you lose access to those instances, regardless of what state the instances are in.

To put instances back into service, use the **as-exit-standby** command in the Auto Scaling CLI, the **aws autoscaling exit-standby** command in the AWS CLI, or the **ExitStandby** API action.

For more information about how to use the standby state with your Auto Scaling groups, see:

- [Updating or Modifying Instances in an Auto Scaling Group](#) (p. 223)
- [Troubleshooting Instances in an Auto Scaling Group](#) (p. 217)

Troubleshooting Instances in an Auto Scaling Group

You have the option of putting an instance that is currently in service into a `Standby` state. One reason for performing this action is because you want to troubleshoot an instance that is not functioning at the level you expect. By putting the instance into a `Standby` state, you can make changes to the instances and then return it to service.

The process described in this section requires that an instance is currently in service (to put more technically, 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. 151).

Troubleshooting Instances Using the Command Line Interface

The following steps demonstrate the general process for troubleshooting an instance that is currently in service. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To troubleshoot an instance that is currently in service using the CLI

1. Run the `as-describe-auto-scaling-instances` command to identify the instance to update.

```
as-describe-auto-scaling-instances
```

This command returns a list of EC2 instances.

```
INSTANCE  i-694c873b  test  us-east-1a  InService  HEALTHY  lc
INSTANCE  i-e116ddb3  test  us-east-1a  InService  HEALTHY  lc
```

2. Move the instance into a `Standby` state using the `as-enter-standby` command.

When you call this command, you must decide if you want to decrement the desired capacity for the Auto Scaling group. If you decide not to decrement the desired capacity, the Auto Scaling group launches additional instances to compensate for the instances moved into the `Standby` state. Auto Scaling automatically increments the desired capacity when you put the instance back in service.

```
as-enter-standby i-e116ddb3 --auto-scaling-group my-asg --decrement-desired-
capacity
```

This command returns a response similar to the following:

```
INSTANCE 0383799c-a411-432e-979b-c8af68222db3 InProgress At 2014-06-06T16:12:28Z instance i-e116ddb3 was moved to standby in response to a user request, shrinking the capacity from 2 to 1.
```

3. Optional. Verify that the instance is in the Standby state using the `as-describe-auto-scaling-instances` command.

```
as-describe-auto-scaling-instances
```

Notice that the status of the instance is now set to Standby.

```
INSTANCE i-694c873b my-asg us-east-1a InService HEALTHY lc
INSTANCE i-e116ddb3 my-asg us-east-1a Standby HEALTHY lc
```

4. Connect to the instance and review logs or run diagnostics as needed.

For more information about how to connect to an Amazon EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Put the instance back in service using the `as-exit-standby` command.

```
as-exit-standby i-e116ddb3 --auto-scaling-group my-asg
```

This command returns a response similar to the following:

```
INSTANCE 94a09ebc-bc7e-44a6-b33d-8ed6f4a652b0 PreInService At 2014-06-06T16:23:00Z instance i-e116ddb3 was moved out of standby in response to a user request, increasing the capacity from 1 to 2.
```

6. Optional. Verify that the instance is back in service using the `as-describe-auto-scaling-instances` command.

```
as-describe-auto-scaling-instances
```

You should now see that the instance is back in service.

```
INSTANCE i-694c873b my-asg us-east-1a InService HEALTHY lc
INSTANCE i-e116ddb3 my-asg us-east-1a InService HEALTHY lc
```

Troubleshooting Using the Query API

The following steps demonstrate the general process for troubleshooting an instance that is currently in service. This example assumes that you already have an Auto Scaling group, *my-asg*, using an existing Auto Scaling launch configuration. For more information about creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To troubleshoot an instance using the Query API

1. Identify the instance to update.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=DescribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A29%3A07.558Z&AUTHPARAMS
```

The following response describes the instances in the Auto Scaling group.

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5b73d709</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-dd70d48f</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-de70d48c</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-df70d48d</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
</DescribeAutoScalingInstancesResponse>
```



```
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>222e5c84-f34a-11e3-bc51-b35178f0274f</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

2. Move the instance into a Standby state using the `EnterStandby` command.

When you call this command, you must decide if you want to decrement the desired capacity for the Auto Scaling group. If you decide not to decrement the desired capacity, the Auto Scaling group launches additional instances to compensate for the instances moved into the `Standby` state. Auto Scaling automatically increments the desired capacity when you put the instance back in service.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&ShouldDecrementDesiredCapacity=true&InstanceIds.member.1=i-5b73d709&Version=2011-01-01&Action=EnterStandby&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A35%3A50.567Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<EnterStandbyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <EnterStandbyResult>
    <Activities>
      <member>
        <ActivityId>462b4bc3-ad3b-4e67-a58d-96cd00f02f9e</ActivityId>
        <Progress>50</Progress>
        <StatusCode>InProgress</StatusCode>
        <StartTime>2014-06-13T22:35:50.884Z</StartTime>
        <Cause>At 2014-06-13T22:35:50Z instance i-5b73d709 was moved to
standby in response to a user request, shrinking the capacity from 4 to
3.</Cause>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <Details>{"Availability Zone":"us-east-1a"}</Details>
        <Description>Moving EC2 instance to Standby: i-5b73d709</Description>
      </member>
    </Activities>
  </EnterStandbyResult>
  <ResponseMetadata>
    <RequestId>l26f2f31-f34b-11e3-bc51-b35178f0274f</RequestId>
  </ResponseMetadata>
</EnterStandbyResponse>
```

3. Optional. Verify that the instance is in the `Standby` state.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=DescribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=Hmac
```

```
SHA256&Timestamp=2014  
-06-13T22%3A38%3A59.818Z&AUTHPARAMS
```

Notice that the status of the instance is now set to Standby.

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon  
aws.com/doc/2011-01-01/">  
  <DescribeAutoScalingInstancesResult>  
    <AutoScalingInstances>  
      <member>  
        <HealthStatus>HEALTHY</HealthStatus>  
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>  
        <AvailabilityZone>us-east-1a</AvailabilityZone>  
        <InstanceId>i-5b73d709</InstanceId>  
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>  
        <LifecycleState>Standby</LifecycleState>  
      </member>  
      <member>  
        <HealthStatus>HEALTHY</HealthStatus>  
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>  
        <AvailabilityZone>us-east-1a</AvailabilityZone>  
        <InstanceId>i-dd70d48f</InstanceId>  
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>  
        <LifecycleState>InService</LifecycleState>  
      </member>  
      <member>  
        <HealthStatus>HEALTHY</HealthStatus>  
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>  
        <AvailabilityZone>us-east-1a</AvailabilityZone>  
        <InstanceId>i-de70d48c</InstanceId>  
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>  
        <LifecycleState>InService</LifecycleState>  
      </member>  
      <member>  
        <HealthStatus>HEALTHY</HealthStatus>  
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>  
        <AvailabilityZone>us-east-1a</AvailabilityZone>  
        <InstanceId>i-df70d48d</InstanceId>  
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>  
        <LifecycleState>InService</LifecycleState>  
      </member>  
    </AutoScalingInstances>  
  </DescribeAutoScalingInstancesResult>  
  <ResponseMetadata>  
    <RequestId>833d0e12-f34b-11e3-a434-7f10009d5849</RequestId>  
  </ResponseMetadata>  
</DescribeAutoScalingInstancesResponse>
```

4. Connect to the instance and review logs or run diagnostics as needed.

For more information about how to connect to an EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Put the instance back in service.

```
http://autoscaling.amazonaws.com/?InstanceIds.member.1=i-5b73d709&AutoScalingGroupName=my-asg&Version=2011-01-01&Action=ExitStandby&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A43%3A53.182Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<ExitStandbyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ExitStandbyResult>
    <Activities>
      <member>
        <ActivityId>dca4efcf-eea6-4844-8064-cablfecd1aa2</ActivityId>
        <Progress>30</Progress>
        <StatusCode>PreInService</StatusCode>
        <StartTime>2014-06-13T22:43:53.523Z</StartTime>
        <Cause>At 2014-06-13T22:43:53Z instance i-5b73d709 was moved out of
standby in response to a user request, increasing the capacity from 3 to
4.</Cause>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <Details>{"Availability Zone":"us-east-1a"}</Details>
        <Description>Moving EC2 instance out of Standby: i-5b73d709</Descrip
tion>
      </member>
    </Activities>
  </ExitStandbyResult>
  <ResponseMetadata>
    <RequestId>321a11c8-f34c-11e3-a434-7f10009d5849</RequestId>
  </ResponseMetadata>
</ExitStandbyResponse>
```

6. Optional. Verify that the instance is back in service.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=De
scribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=Hmac
SHA256&Timestamp=2014
-06-13T22%3A50%3A54.080Z&AUTHPARAMS
```

You should now see that the instance is back in service.

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5b73d709</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
</DescribeAutoScalingInstancesResponse>
```

```
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-dd70d48f</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-de70d48c</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-df70d48d</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>2cf127d4-f34d-11e3-a434-7f10009d5849</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

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 change any instances currently in service. You can update these instances by putting them into a `Standby` state. This state allows you to update the software for the instance, and then put the instance back in service.

Updating an Instance Using the Command Line

The following steps demonstrate the general process for updating an instance that is currently in service. This example assumes that you already have an Auto Scaling group, `my-asg`, using an existing Auto Scaling launch configuration. For more information on creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To update software on an instance using the CLI

1. Run the `as-describe-auto-scaling-instances` command to identify the instance to update.

```
as-describe-auto-scaling-instances
```

This command returns a list of EC2 instances.

```
INSTANCE  i-5b73d709  my-asg  us-east-1a  InService  HEALTHY  my-asg
INSTANCE  i-dd70d48f  my-asg  us-east-1a  InService  HEALTHY  my-asg
INSTANCE  i-de70d48c  my-asg  us-east-1a  InService  HEALTHY  my-asg
INSTANCE  i-df70d48d  my-asg  us-east-1a  InService  HEALTHY  my-asg
```

2. Move the instance into a `Standby` state using the `as-enter-standby` command.

When you call this command, you must decide if you also want to decrement the desired capacity for the Auto Scaling group. If you decide not to decrement the desired capacity, the Auto Scaling group launches additional instances to compensate for the instances moved into the `Standby` state. Auto Scaling automatically increments the desired capacity when you put the instance back in service.

```
as-enter-standby --instances i-5b73d709 --auto-scaling-group my-asg --
decrement-desired-capacity
```

This command returns a response similar to the following:

```
INSTANCE  309f9e29-4f24-44f7-bbd0-2d8a54fa3e39  InProgress  At 2014-06-
13T22:21:25Z instance i-5b73d709 was moved to standby in response to
a user request, shrinking the capacity from 4 to 3.
```

3. Optional. Verify that the instance is in `Standby` using the `as-describe-auto-scaling-instances` command.

```
as-describe-auto-scaling-instances
```

Notice that the status of the instance is now set to `Standby`.

```
INSTANCE  i-5b73d709  my-asg  us-east-1a  Standby  HEALTHY  my-asg
INSTANCE  i-dd70d48f  my-asg  us-east-1a  InService  HEALTHY  my-asg
INSTANCE  i-de70d48c  my-asg  us-east-1a  InService  HEALTHY  my-asg
INSTANCE  i-df70d48d  my-asg  us-east-1a  InService  HEALTHY  my-asg
```

4. Connect to the instance and update the software as needed.

For more information about how to connect to an Amazon EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Put the instance back in service using the `as-exit-standby` command.

```
as-exit-standby --instances i-5b73d709 --auto-scaling-group my-asg
```

This command returns a response similar to the following:

```
INSTANCE a31ac4ce-a144-488d-b112-23431e4f6fd2 PreInService At 2014-06-13T22:24:49Z instance i-5b73d709 was moved out of standby in response to a user request, increasing the capacity from 3 to 4.
```

6. Optional. Verify that the instance is back in service using the `as-describe-auto-scaling-instances` command.

```
as-describe-auto-scaling-instances
```

You should now see that the instance is back in service.

```
INSTANCE i-5b73d709 my-asg us-east-1a InService HEALTHY my-asg
INSTANCE i-dd70d48f my-asg us-east-1a InService HEALTHY my-asg
INSTANCE i-de70d48c my-asg us-east-1a InService HEALTHY my-asg
INSTANCE i-df70d48d my-asg us-east-1a InService HEALTHY my-asg
```

Updating an Instance Using the Query API

The following steps demonstrate the general process for updating an instance that is currently in service using the Query API. This example assumes that you already have an Auto Scaling group, *my-asg*, using an existing Auto Scaling launch configuration. For more information on creating launch configurations and Auto Scaling groups, see [Getting Started with Auto Scaling](#).

To update software on an instance using the Query API

1. Identify the instance you want to update.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=DescribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A29%3A07.558Z&AUTHPARAMS
```

The following response describes the instances in the Auto Scaling group:

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5b73d709</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
```

```
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-dd70d48f</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-de70d48c</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
<member>
  <HealthStatus>HEALTHY</HealthStatus>
  <AutoScalingGroupName>my-asg</AutoScalingGroupName>
  <AvailabilityZone>us-east-1a</AvailabilityZone>
  <InstanceId>i-df70d48d</InstanceId>
  <LaunchConfigurationName>my-asg</LaunchConfigurationName>
  <LifecycleState>InService</LifecycleState>
</member>
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>222e5c84-f34a-11e3-bc51-b35178f0274f</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

2. Move the instance into a Standby state using the `EnterStandby` command.

When you call this command, you must decide if you want to decrement the desired capacity for the Auto Scaling group. If you decide not to decrement the desired capacity, the Auto Scaling group launches additional instances to compensate for the instances moved into the `Standby` state. Auto Scaling automatically increments the desired capacity when you put the instance back in service.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=my-asg&ShouldDecrementDesiredCapacity=true&InstanceIds.member.1=i-5b73d709&Version=2011-01-01&Action=EnterStandby&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A35%3A50.567Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<EnterStandbyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <EnterStandbyResult>
    <Activities>
      <member>
        <ActivityId>462b4bc3-ad3b-4e67-a58d-96cd00f02f9e</ActivityId>
        <Progress>50</Progress>
        <StatusCode>InProgress</StatusCode>
      </member>
    </Activities>
  </EnterStandbyResult>
</EnterStandbyResponse>
```

```
<StartTime>2014-06-13T22:35:50.884Z</StartTime>
<Cause>At 2014-06-13T22:35:50Z instance i-5b73d709 was moved to
standby in response to a user request, shrinking the capacity from 4 to
3.</Cause>
<AutoScalingGroupName>my-asg</AutoScalingGroupName>
<Details>{"Availability Zone": "us-east-1a"}</Details>
<Description>Moving EC2 instance to Standby: i-5b73d709</Description>

</member>
</Activities>
</EnterStandbyResult>
<ResponseMetadata>
  <RequestId>l26f2f31-f34b-11e3-bc51-b35178f0274f</RequestId>
</ResponseMetadata>
</EnterStandbyResponse>
```

3. Optional. Verify that the instance is in the Standby state.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=De
scribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=Hmac
SHA256&Timestamp=2014
-06-13T22%3A38%3A59.818Z&AUTHPARAMS
```

Notice that the status of the instance is now set to Standby.

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5b73d709</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>Standby</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-dd70d48f</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-de70d48c</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
</DescribeAutoScalingInstancesResponse>
```



```
<HealthStatus>HEALTHY</HealthStatus>
<AutoScalingGroupName>my-asg</AutoScalingGroupName>
<AvailabilityZone>us-east-1a</AvailabilityZone>
<InstanceId>i-df70d48d</InstanceId>
<LaunchConfigurationName>my-asg</LaunchConfigurationName>
<LifecycleState>InService</LifecycleState>
</member>
</AutoScalingInstances>
</DescribeAutoScalingInstancesResult>
<ResponseMetadata>
  <RequestId>833d0e12-f34b-11e3-a434-7f10009d5849</RequestId>
</ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

4. Connect to the instance and update the software as needed.

For more information about how to connect to an Amazon EC2 instance, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Put the instance back in service.

```
http://autoscaling.amazonaws.com/?InstanceId.member.1=i-5b73d709&AutoScalingGroupName=my-asg&Version=2011-01-01&Action=ExitStandby&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A43%3A53.182Z&AUTHPARAMS
```

Auto Scaling returns a response similar to the following:

```
<ExitStandbyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ExitStandbyResult>
    <Activities>
      <member>
        <ActivityId>dca4efcf-eea6-4844-8064-cab1fecdlaa2</ActivityId>
        <Progress>30</Progress>
        <StatusCode>PreInService</StatusCode>
        <StartTime>2014-06-13T22:43:53.523Z</StartTime>
        <Cause>At 2014-06-13T22:43:53Z instance i-5b73d709 was moved out of
standby in response to a user request, increasing the capacity from 3 to
4.</Cause>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <Details>{"Availability Zone":"us-east-1a"}</Details>
        <Description>Moving EC2 instance out of Standby: i-5b73d709</Description>
      </member>
    </Activities>
  </ExitStandbyResult>
  <ResponseMetadata>
    <RequestId>321a11c8-f34c-11e3-a434-7f10009d5849</RequestId>
  </ResponseMetadata>
</ExitStandbyResponse>
```

6. Optional. Verify that the instance is back in service.

```
http://autoscaling.amazonaws.com/?MaxRecords=20&Version=2011-01-01&Action=DescribeAutoScalingInstances&SignatureVersion=2&SignatureMethod=HmacSHA256&Timestamp=2014-06-13T22%3A50%3A54.080Z&AUTHPARAMS
```

You should now see that the instance is back in service.

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-5b73d709</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-dd70d48f</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-de70d48c</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-df70d48d</InstanceId>
        <LaunchConfigurationName>my-asg</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
  <ResponseMetadata>
    <RequestId>2cf127d4-f34d-11e3-a434-7f10009d5849</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

Suspend and Resume Auto Scaling Process

Auto Scaling allows 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.

There are two primary Auto Scaling process types — `Launch` and `Terminate`. The `Launch` process creates a new EC2 instance for an Auto Scaling group, and the `Terminate` process removes an existing Amazon EC2 instance.

The remaining Auto Scaling process types relate to specific Auto Scaling features:

- `AddToLoadBalancer`
- `AlarmNotification`
- `AZRebalance`
- `HealthCheck`
- `ReplaceUnhealthy`
- `ScheduledActions`

If you suspend `Launch` or `Terminate`, all other process types are affected to varying degrees. The following descriptions discuss how each process type is affected by a suspension of a `Launch` or a `Terminate` process.

The `AddToLoadBalancer` process type adds instances to the load balancer when the instances are launched. If you suspend this process, Auto Scaling launches the instances but does not add them to the load balancer. If you resume the `AddToLoadBalancer` process, Auto Scaling also resumes adding new instances to the load balancer when they are launched. However, Auto Scaling does not add running instances that were launched while the process was suspended; those instances must be added manually using the [RegisterInstancesWithLoadBalancer](#) Elastic Load Balancing API action or the `elb-register-instances-with-lb` Elastic Load Balancing command.

The `AlarmNotification` process type accepts notifications from CloudWatch alarms that are associated with the Auto Scaling group. If you suspend the `AlarmNotification` process type, Auto Scaling does not automatically execute scaling policies that would be triggered by alarms.

Although the `AlarmNotification` process type is not directly affected by a suspension of `Launch` or `Terminate`, alarm notifications are often used to signal that a change in the size of the Auto Scaling group is warranted. If you suspend `Launch` or `Terminate` process, Auto Scaling might not be able to implement the alarm's associated policy.

The `AZRebalance` process type seeks to maintain a balanced number of instances across Availability Zones within a 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 application instances evenly across all of the designated Availability Zones.

If you suspend the `Launch` process, the `AZRebalance` process neither launches new instances nor terminates existing instances. This is because the `AZRebalance` process terminates existing instances only after launching the replacement instances.

If you suspend the `Terminate` process, the `AZRebalance` process can cause your Auto Scaling group to grow up to ten percent larger than the maximum size. This is because Auto Scaling allows groups to grow temporarily larger than the maximum size during rebalancing activities. If Auto Scaling cannot ter-

minate instances, your Auto Scaling group could remain up to ten percent larger than the maximum size until you resume the terminate process type.

The `HealthCheck` process type checks the health of the instances. Auto Scaling marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing informs Auto Scaling that the instance is unhealthy. The `HealthCheck` process can override the health status of an instance that you set with [SetInstanceHealth](#) API action or the `as-set-instance-health` command.

The `ReplaceUnhealthy` process terminates instances that are marked as unhealthy and subsequently creates new instances to replace them. This process calls both of the primary process type — first `Terminate` and then `Launch`.

The `HealthCheck` process works in conjunction with the `ReplaceUnhealthy` process to provide health check functionality. If you suspend either the `Launch` or the `Terminate` process, the `ReplaceUnhealthy` process will not function properly.

The `ScheduledActions` process type performs scheduled actions that you create with either the [PutScheduledUpdateGroupAction](#) API action or the `as-put-scheduled-update-group-action` command. Scheduled actions often involve launching new instances or terminating existing instances. If you suspend either the `Launch` or the `Terminate` process, your scheduled actions might not function as expected.

Auto Scaling might, at times, 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 no running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time in launching any instances.

Auto Scaling allows you to resume both, processes suspended for administrative reasons and processes suspended following a suspension of `Launch` or `Terminate`.

The following procedures walk you through the process of suspending all scaling activities on your Auto Scaling group to investigate a configuration problem with your Amazon EC2 application. After you conclude the investigation, you can resume scaling activities on the Auto Scaling group.

You can suspend and then resume the scaling process on your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. For information about installing the command line interface, see [Install the Auto Scaling CLI](#) (p. 16). For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs](#) (p. 23).

Topics

- [Suspend Processes Using the Command Line Interface](#) (p. 231)
- [Suspend Processes Using the Query API](#) (p. 232)

Suspend Processes Using the Command Line Interface

This example assumes that you have an Amazon EC2 application running within a single Auto Scaling group named `MyAutoScalingGroup`.

To suspend and resume processes on an Auto Scaling group

1. Use the `as-suspend-processes` command by specifying the following value:
 - Auto Scaling group name = `MyAutoScalingGroup`

Your command should look similar to the following example:

```
PROMPT>as-suspend-processes MyAutoScalingGroup
```

Auto Scaling returns the following:

```
OK-Processes Suspended
```

2. After concluding your investigation, use the `as-resume-processes` command by specifying the following value:

- Auto Scaling group name = `MyAutoScalingGroup`

Your command should look similar to the following example:

```
PROMPT>as-resume-processes MyAutoScalingGroup
```

Auto Scaling returns the following:

```
OK-Processes Resumed
```

Your Amazon EC2 application has resumed normal Auto Scaling activities.

Suspend Processes Using the Query API

This example assumes that you have an Amazon EC2 application running within a single Auto Scaling group named `MyAutoScalingGroup`.

To suspend and then resume scaling processes on an Auto Scaling group

1. Call the [SuspendProcesses](#) action with the following parameter:
 - `AutoScalingGroupName` = `MyAutoScalingGroup`
2. After concluding your investigation, call the [ResumeProcesses](#) action with the following parameter:
 - `AutoScalingGroupName` = `MyAutoScalingGroup`

Your Amazon EC2 application resumes normal Auto Scaling activities.

Shut Down Your Auto Scaling Process

The Auto Scaling process can be completely shut down by following these steps:

- Update your Auto Scaling group by specifying 0 for your maximum, minimum, and desired (if defined) number of instances.
- Delete the Auto Scaling group.
- [optional] Delete the launch configuration.

- [optional] Delete the load balancer.
- [optional] Delete the CloudWatch alarms.

Topics

- [Shutting Down Using the Command Line Interface \(p. 233\)](#)
- [Shutting Down Using the Query API \(p. 236\)](#)

The following sections step you through the process of completely shutting down the Auto Scaling process for your Auto Scaling group.

You can shut down the Auto Scaling process for your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. For information about installing the command line interface, see [Install the Auto Scaling CLI \(p. 16\)](#). If you have used Elastic Load Balancing and CloudWatch alarms with your Auto Scaling group and are planning to shut down your load balancer and the alarms, make sure to install the CLI for those services as well. For information about the Elastic Load Balancing CLI, see [Installing the Command Line Interface](#) in the *Elastic Load Balancing Developer Guide*. For information about the CloudWatch CLI, see [Command Line Tools](#) in the *Amazon CloudWatch Developer Guide*. For information about creating a Query request, see [Use Query Requests to Call Auto Scaling APIs \(p. 23\)](#).

Shutting Down Using the Command Line Interface

You can use the CLI to shut down the scaling process completely for your Auto Scaling group.

Delete Your Auto Scaling Group

You can delete your Auto Scaling group if the group has no running instances. To ensure that your Auto Scaling group has no running instances, update your Auto Scaling group by specifying the minimum size and the maximum size as zero instances.

To delete your Auto Scaling group

1. Update your Auto Scaling group
 - a. Use the `as-update-auto-scaling-group` command and specify the following values:
 - Auto Scaling group name = `my-test-asg`
 - Max size = 0
 - Min size = 0

Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg --max-size 0 --min-size 0
```

- b. You should get a confirmation similar to the following example:

```
OK-Updated AutoScalingGroup
```

2. Verify that your Auto Scaling group has no running instances
 - a. Use the `as-describe-auto-scaling-groups` command and specify the following value:

- Auto Scaling group name = `my-test-asg`

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg
```

- b. Auto Scaling might report that the state of your instances is `Terminating` because the termination process can take a few minutes.

After the termination process completes, you should get a confirmation similar to the following example:

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	LOAD-BALANCERS	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-east-1a					my-
test-asg-loadbalancer	0	0	0					Default

3. Delete your Auto Scaling group.

- a. Use the `as-delete-auto-scaling-group` command and specify the following value:

- Auto Scaling group name = `my-test-asg`

Your command should look similar to the following example:

```
as-delete-auto-scaling-group my-test-asg
```

- b. Auto Scaling returns the following question:

```
Are you sure you want to delete this AutoScalingGroup? [Ny]
```

- c. Enter `y` to delete the Auto Scaling group.

- d. Auto Scaling returns the following response:

```
OK-Deleted AutoScalingGroup
```

(Optional) Delete the Launch Configuration Associated with Your Auto Scaling Group

Skip this step if you are planning on using the launch configuration later to launch Auto Scaling groups.

To delete the launch configuration associated with your Auto Scaling group

1. Use the `as-delete-launch-config` command and specify the following value:

- Launch configuration name = `my-test-lc`

Your command should look similar to the following example:

```
as-delete-launch-config my-test-lc
```

2. Auto Scaling returns the following:

```
Are you sure you want to delete this launch configuration? [Ny]
```

3. Enter `y` to delete the launch configuration.
4. Auto Scaling returns the following:

```
OK-Deleted launch configuration
```

(Optional) Delete the Load Balancer

Skip this step if your Auto Scaling group is not registered with your Elastic Load Balancing load balancer or you do not wish to delete your load balancer.

To delete the load balancer

1. Use the Elastic Load Balancing command `elb-delete-lbs` and specify the following value:
 - Load balancer name = `my-test-asg-loadbalancer`

Your command should look similar to the following example:

```
elb-delete-lb my-test-asg-loadbalancer
```

2. Elastic Load Balancing returns the following:

```
Warning: Deleting a LoadBalancer can lead to service disruption to any cus  
tomers connected to the load balancer. Are you sure you want to delete this  
load balancer? [Ny]
```

3. Enter `y` to delete the load balancer.
4. Elastic Load Balancing returns the following:

```
OK-Deleting LoadBalancer
```

(Optional) Delete CloudWatch Alarms

Skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you do not wish to delete the CloudWatch alarms.

To delete CloudWatch alarms

1. Use the CloudWatch command `mon-delete-alarms` and specify the following value:
 - Alarm names = *AddCapacity, RemoveCapacity*

Your command should look similar to the following example:

```
mon-delete-alarms AddCapacity, RemoveCapacity
```

2. CloudWatch returns the following:

```
Are you sure you want to delete these Alarms? [Ny]y
```

3. Enter `y` to delete the alarms.
4. You should get confirmation similar to the following example:

```
OK-Deleting Alarms
```

Shutting Down Using the Query API

You can use the Query API to completely shut down the scaling process for your Auto Scaling group.

Delete Your Auto Scaling Group

You can delete your Auto Scaling group if the group has no running instances. To ensure that your Auto Scaling group has no running instances, update your Auto Scaling group by specifying the minimum size and the maximum size as zero instances.

To delete your Auto Scaling group

1. Update your Auto Scaling group.
 - a. Call the [UpdateAutoScalingGroup](#) action and specify the following parameters:
 - AutoScalingGroupName = `my-test-asg`
 - MaxSize = 0
 - MinSize = 0

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&MinSize=0
&MaxSize=0
&Version=2011-01-01
&Action=UpdateAutoScalingGroup
&AUTHPARAM
```

- b. If your request was successful, you should get a confirmation similar to the following example:

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>0d361eb7-9665-11e2-80ec-8d4dEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

2. Verify that your Auto Scaling group has no running instances.
 - a. Call the [DescribeAutoScalingGroups](#) action and specify the following parameter:
 - AutoScalingGroupName = my-test-asg

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-
test-asg
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

- b. Auto Scaling might report that the state of your instances is `Terminating` because the termination process can take a few minutes.

After the termination process completes, you should get a confirmation similar to the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-03-17T03:54:14.210Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances/>
        <DesiredCapacity>0</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames>
          <member>my-test-asg-loadbalancer</member>
        </LoadBalancerNames>
        <MinSize>0</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
```

```
1:803981987763:autoScalingGroup:23639c1d-6703-4d8e-bd04-
1428a16e0770:autoScalingGroupName/my-test-asg</AutoScalingGroupARN>
  <TerminationPolicies>
    <member>Default</member>
  </TerminationPolicies>
  <MaxSize>0</MaxSize>
</member>
</AutoScalingGroups>
</DescribeAutoScalingGroupsResult>
<ResponseMetadata>
  <RequestId>57d72661-9664-11e2-b1f1-2f998EXAMPLE</RequestId>
</ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

3. Delete your Auto Scaling group.

a. Call the [DeleteAutoScalingGroup](#) action and specify the following parameter:

- AutoScalingGroupName = my-test-asg

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&ForceDelete=false
&Version=2011-01-01
&Action=DeleteAutoScalingGroup
&AUTHPARAM
```

b. If your request was successful, you should get a confirmation similar to the following example:

```
<DeleteAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>70a76d42-9665-11e2-9fdf-211deEXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteAutoScalingGroupResponse>
```

(Optional) Delete the Launch Configuration Associated with Your Auto Scaling Group

Skip this step if you are planning on using the launch configuration later to launch Auto Scaling groups.

To delete the launch configuration associated with your Auto Scaling group

1. Call the [DeleteLaunchConfiguration](#) action and specify the following parameter:

- LaunchConfigurationName = my-test-lc

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?LaunchConfigurationName=my-test-lc
&Version=2011-01-01
&Action=DeleteLaunchConfiguration
&AUTHPARAM
```

2. If your request was successful, you should get a confirmation similar to the following example:

```
<DeleteLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7347261f-97df-11e2-8756-35eEXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteLaunchConfigurationResponse>
```

(Optional) Delete Load Balancer

Skip this step if your Auto Scaling group is not registered with a Elastic Load Balancing load balancer or you do not wish to delete your load balancer.

To delete the load balancer

1. Call the [DeleteLoadBalancer](#) Elastic Load Balancing action and specify the following parameter:
 - `LoadBalancerName = my-test-asg-loadbalancer`

Your request should look similar to the following example:

```
https://elasticloadbalancing.amazonaws.com/?LoadBalancerName=my-test-asg-
loadbalancer
&Version=2012-06-01
&Action=DeleteLoadBalancer
&AUTHPARAMS
```

2. If your request was successful, you should get a confirmation similar to the following example:

```
<DeleteLoadBalancerResponse xmlns="http://elasticloadbalancing.amazon
aws.com/doc/2012-06-01/">
  <DeleteLoadBalancerResult/>
  <ResponseMetadata>
    <RequestId>806af50e-97ee-11e2-b66d-8133EXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteLoadBalancerResponse>
```

(Optional) Delete CloudWatch Alarms

Skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you do not wish to delete the CloudWatch alarms.

To delete CloudWatch alarms

1. Call the `DeleteAlarms` CloudWatch action and specify the following parameters:

- `AlarmNames.member.1` = *AddCapacity*
- `AlarmNames.member.2` = *RemoveCapacity*

Your request should look similar to the following example:

```
https://monitoring.us-east-1.amazonaws.com/?AlarmNames.member.1=AddCapacity
&AlarmNames.member.2=RemoveCapacity
&Version=2010-08-01
&Action=DeleteAlarms
&AUTHPARAMS
```

2. If your request was successful, you should get confirmation similar to the following example:

```
<DeleteAlarmsResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-01/">
  <ResponseMetadata>
    <RequestId>66c9f789-adc9-11e2-981d-c7104EXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteAlarmsResponse>
```

Monitoring Your Auto Scaling Instances

Topics

- [Amazon CloudWatch Alarms \(p. 241\)](#)
- [Activating Detailed Instance Monitoring for Auto Scaling \(p. 242\)](#)
- [Activating Basic Instance Monitoring for Auto Scaling \(p. 243\)](#)
- [Auto Scaling Group Metrics \(p. 243\)](#)
- [Health Checks \(p. 244\)](#)
- [Getting Notifications When Your Auto Scaling Group Changes \(p. 248\)](#)
- [Logging Auto Scaling API Calls By Using AWS CloudTrail \(p. 254\)](#)

This section discusses the metrics that Auto Scaling instances send 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.

Note

Selecting detailed monitoring is a prerequisite for the collection of Auto Scaling group metrics. For more information, see [Auto Scaling Group Metrics \(p. 243\)](#).

The following sections describe how to enable detailed monitoring or basic monitoring.

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. 243).
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 \(p. 241\)](#).
2. Call `EnableMetricsCollection`, which is part of the Auto Scaling Query API. 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 unhealthy if the calls to the Amazon EC2 action [DescribeInstanceStatus](#) returns any other state other than `running`, the system status shows `impaired`, or the calls to Elastic Load Balancing action [DescribeInstanceHealth](#) returns `OutOfService` in the instance state field.

After an instance is marked unhealthy because of an Amazon EC2 or Elastic Load Balancing health check, it is scheduled for replacement.

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 about the Auto Scaling health check, see [Maintaining a Fixed Number of Running EC2 Instances](#) (p. 53).

For information about adding the Elastic Load Balancing health check, see [Add an Elastic Load Balancing Health Check to your Auto Scaling Group](#) (p. 195). For information about adding a customized health check, see [Configure the Health State of An Instance](#) (p. 245).

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*.

Configure the Health State of An Instance

Auto Scaling ensures that the EC2 instances within the Auto Scaling group are running and in good shape by periodically performing health checks on the instances. When Auto Scaling determines that an instance is unhealthy, it terminates that instance and launches a new one. This helps in maintaining the number of running instances at the minimum number (or desired number, if specified) that you defined.

By default, your Auto Scaling group determines the health state of each instance by periodically checking the results of EC2 instance status checks. If you have associated your Auto Scaling group with an Elastic Load Balancing 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 the EC2 instance status checks and the Elastic Load Balancing instance health checks.

If you have your own health check system, you can use the information from your health check system to set the health state of the instances in the Auto Scaling group.

The following sections step you through the process for explicitly specifying the health state of an instance in your Auto Scaling group. You can use either the Auto Scaling command line interface (CLI) or the Query API to complete this task.

Configuring the Health State Using the Command Line Interface

You can use the Auto Scaling CLI to configure the health state of an instance and then verify the instance's health state.

To configure the health state of an instance using the CLI

1. Use the `as-set-instance-health` command by specifying the following values:
 - Instance ID = `i-123abc45d`
 - Health Status = `Unhealthy`

Your command should look similar to the following example:

```
as-set-instance-health i-123abc45d --status Unhealthy
```

You should get a confirmation similar to the following example:

```
OK-Instance Health Set
```

2. Use the `as-describe-auto-scaling-groups` command to verify if the instance is marked Unhealthy. You can optionally specify the following value:

- Auto Scaling group name = *my-test-asg*

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg
```

You should get a response that includes the details of the instances in the Auto Scaling group, as in the following example:

```
.....  
INSTANCE  i-123abc45d us-east-1a  Terminating  Unhealthy  my-test-lc  
.....
```

Auto Scaling has marked the instance Unhealthy and has started terminating the instance.

Configuring the Health State Using the Query API

You can use the Auto Scaling Query API to configure the health state of an instance and then verify the instance's health state.

To configure the health state of an instance using the Query API

1. Call the [SetInstanceHealth](#) action by specifying the following parameters:

- InstanceId = *i-123abc45d*
- HealthStatus = Unhealthy

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?InstanceId=i-123abc45d  
&ShouldRespectGracePeriod=true  
&HealthStatus=Unhealthy  
&Version=2011-01-01  
&Action=SetInstanceHealth  
&AUTHPARAMS
```

If your request was successful, you should get a confirmation that looks similar to the following example:

```
<SetInstanceHealthResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>a3603249-ad22-11e2-bada-31b1bEXAMPLE</RequestId>
  </ResponseMetadata>
</SetInstanceHealthResponse>
```

2. Call the [DescribeAutoScalingGroups](#) action to verify if the instance is marked Unhealthy. You can optionally specify the following parameter:
 - AutoScalingGroupNames.member.1 = *my-test-asg*

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-asg
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

The response includes details of the instances in the Auto Scaling group, as in the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Unhealthy</HealthStatus>
            <AvailabilityZone>us-east-1c</AvailabilityZone>
            <InstanceId>i-i-123abc45d</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>Terminating</LifecycleState>
          </member>
          .....
          .....
        </Instances>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
</DescribeAutoScalingGroupsResponse>
```

Auto Scaling has marked the instance Unhealthy and has started terminating the instance.

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, 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*.

This section goes through the following steps for configuring your Auto Scaling group to send email notifications to the email address you specify, whenever your Auto Scaling group changes. Before you begin, make sure that you have created an Auto Scaling group.

Topics

- [Configure Amazon SNS](#) (p. 248)
- [Configure your Auto Scaling Group to Send Notifications](#) (p. 249)
- [Test the Notification Configuration](#) (p. 251)
- [Verify That You Received Notification of the Scaling Event](#) (p. 253)
- [Delete the Notification Configuration](#) (p. 254)

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 Amazon 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 the [Create a Topic](#) section in the *Amazon Simple Notification Service Developer Guide*.

Make a note of the topic Amazon Resource Name (ARN). You need it for the next step to subscribe to the topic and later if you are using the Auto Scaling CLI or the Query API to configure your Auto Scaling group to use the topic.

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.

Follow the instructions in the [Subscribe to a Topic](#).

Note

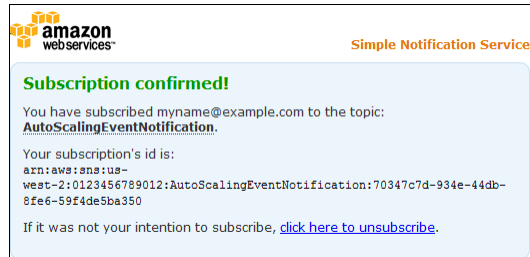
For information about other endpoints you can use to receive notifications, see the [Amazon Simple Notification Service Developer Guide](#).

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 similar to the following example:



Amazon SNS is now configured to receive notifications and send the notification as an email to the specified email address.

Configure your Auto Scaling Group to Send Notifications

In this section, you configure Auto Scaling group to send notifications to Amazon SNS when a scaling event, such as launching instances or terminate instances, takes place. Amazon SNS sends a notification to the email address that you specified in the previous step with information about the instances.

Auto Scaling supports sending Amazon SNS notifications when the following events occur:

Notification type	Event
autoscaling:EC2_INSTANCE_LAUNCH	Successful instance launch
autoscaling:EC2_INSTANCE_LAUNCH_ERROR	Failed instance launch
autoscaling:EC2_INSTANCE_TERMINATE	Successful instance termination
autoscaling:EC2_INSTANCE_TERMINATE_ERROR	Failed instance termination
autoscaling:TEST_NOTIFICATION	Validated a configured Amazon SNS topic (as a result of calling the PutNotificationConfiguration action)

When you configure your Auto Scaling group to send email notifications, you must specify the notification types that the Auto Scaling group must use to format the content of the email. 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 to the email account that you specify. The email contains the details of the terminated instance, such as the instance ID and the reason why the instance was terminated.

You can configure your Auto Scaling group to send notifications using the AWS Management Console, the Auto Scaling CLI, the AWS CLI, or the Query API.

Topics

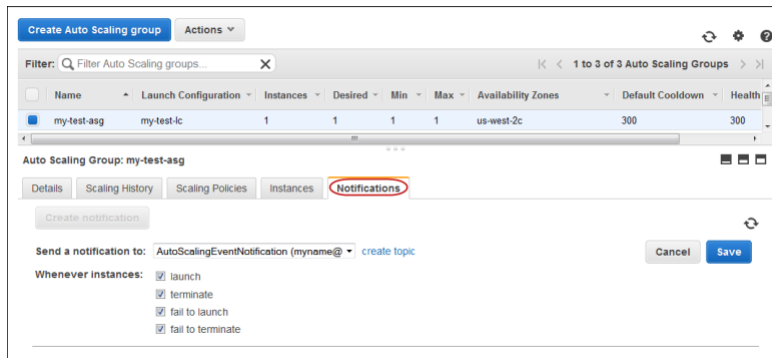
- [Configure Notifications Using the Console \(p. 250\)](#)
- [Configure Notifications Using the Command Line Interface \(p. 250\)](#)

Configure Notifications Using the Console

Use the Auto Scaling console to configure Amazon SNS notifications for your existing Auto Scaling group.

To configure Amazon SNS notifications for your Auto Scaling group using the console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group (*my-test-asg*) 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, perform the following actions:
 - a. Click the **Send a notification to:** field and select your topic.
 - b. In the **Whenever instances:** event list, select the events to send the notifications for.



- c. Click **Save**.

Configure Notifications Using the Command Line Interface

After you've created your Amazon SNS topic and you have the ARN, you are ready to set up the notification configuration using the Auto Scaling CLI. If you have not yet installed the CLI, follow the instructions in [Install the Auto Scaling CLI \(p. 16\)](#) to do so.

To configure Amazon SNS notifications for your Auto Scaling group

1. Use the `as-put-notification-configuration` command and specify the following values:
 - Auto Scaling group name: *my-test-asg*
 - ARN: *ARN-placeholder*

Note

ARNs are unique identifiers for AWS resources. Replace the ARN placeholder with your ARN.

- Notification types: `autoscaling:EC2_Instance_Launch`, `autoscaling:EC2_Instance_Terminate`

Your command should look similar to the following example:

```
as-put-notification-configuration my-test-asg --topic-arn your-ARN-here -  
-notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN  
STANCE_TERMINATE
```

2. Auto Scaling returns the following:

```
OK-Put Notification Configuration
```

3. Use `as-describe-notification-configurations` command and specify the following value to verify that your Auto Scaling group is configured to send notifications.

- Auto Scaling group name: `my-test-asg`

Note

If you specify the Auto Scaling group, this command returns a full list of all notification configurations for the Auto Scaling group listed. If you don't provide an Auto Scaling group name, the service returns the full details of all Auto Scaling groups. The command also returns a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the `next-token` argument.

Your command should look similar to the following example:

```
as-describe-notification-configurations --auto-scaling-groups my-test-asg  
-headers
```

Auto Scaling returns the following:

```
NOTIFICATION-CONFIG GROUP-NAME TOPIC-ARN NOTIFICATION-TYPE-NAME  
NOTIFICATION-CONFIG my-test-asg your-ARN-here autoscaling:EC2_IN  
STANCE_LAUNCH  
NOTIFICATION-CONFIG my-test-asg your-ARN-here autoscaling:EC2_INSTANCE_TER  
MINATE
```

You have confirmed that you have a notification configuration set up for the `my-test-asg` Auto Scaling group.

Test the Notification Configuration

To cause the changes that generate notifications, update the Auto Scaling group by changing the desired capacity of the `my-test-asg` Auto Scaling group from 1 instance to 2 instances. When Auto Scaling launches the EC2 instance to the new desired capacity, Amazon SNS sends an email notification.

You can change the desired capacity of your Auto Scaling group using the AWS Management Console, the Auto Scaling CLI, the AWS CLI, or the Query API.

This section covers instructions using the console or the Auto Scaling CLI.

Changing Capacity Using the Console

Use the Auto Scaling console to change the desired capacity of your Auto Scaling group.

To change the desired capacity using the console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group (*my-test-asg*) from the list.
4. The bottom pane displays the details of your Auto Scaling group. Select the **Details** tab.
5. Click **Edit**.
6. In the **Desired** field, enter 2 and click **Save**.

Within a few minutes of changing the desired capacity of your Auto Scaling group, you should get an email notification that the instance for *my-test-asg* was launched.

Changing Capacity Using the Command Line Interface

In this example, you use the CLI to change the desired capacity of the Auto Scaling group.

To change the desired capacity using the CLI

1. Use the `as-set-desired-capacity` command by specifying the following values:
 - Auto Scaling group name: *my-test-asg*
 - Desired capacity: *2*
 - [optional]`honor-cooldown`

Note

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 \(p. 46\)](#).

2. Your command should look similar to the following example:

```
as-set-desired-capacity my-test-asg --desired-capacity 2
```

3. Auto Scaling returns the following:

```
OK-Desired Capacity Set
```

Within a few minutes of calling `as-set-desired-capacity`, you should get an email notification that the instance for *my-test-asg* was launched.

Verify That You Received Notification of the Scaling Event

Check your email for a message from Amazon SNS and open the email. The email content should be similar to the following example:

```
Subject: AWS Auto Scaling
Time: 2014-01-11T18:31:44.071Z
RequestID: 807226-3815-4b15-ba4b-215f6ca2b68
EventID: auto-scaling-ec2-notification-launch
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:2012745678901:autoScalingGroup:1:20140110-275e-4f13-b72e-ae08213d08:autoScalingGroupName/my-test-asg
AutoScalingGroupMinSize: 1
AutoScalingGroupMaxSize: 3
Cause: At 2014-01-11T18:30:34Z a user request update of AutoScalingGroup constraints to min: 1, max: 3, desired: 2 changing the desired capacity from 1 to 2. At 2014-01-11T18:30:43Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.
StartTime: 2014-01-11T18:30:43.552Z
EndTime: 2014-01-11T18:31:44.071Z
StatusCode:InProgress
StatusMessage:
Progress:50
EC2InstanceID:i-0998ded1
Details:{"AvailabilityZone":"us-west-2"}
```

After you check your email and confirm that you have received notification of a scaling event for your Auto Scaling group, you can confirm the scaling event by looking at the description of your Auto Scaling group.

To verify that your Auto Scaling group has launched new instance using the console

1. In the console, go back to your Auto Scaling group page.
2. Make sure your Auto Scaling group `my-test-asg` is selected and then in the bottom description pane, click **Scaling History**.
3. The **Status** column lets you know the current status of your instance. Click refresh button to see the status of your new instance change to **Successful**, indicating that your Auto Scaling group has successfully launched a new instance.
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.

Check the **Instance ID** of the new instance. It matches the instance ID mentioned in the **EC2InstanceID**: field in the notification email.

To verify that your Auto Scaling group has launched new instance using the CLI

1. Use the `as-describe-auto-scaling-groups` command by specifying the name of your Auto Scaling group `my-test-asg` to confirm that the size of your Auto Scaling group has changed.

Your command should look similar to the following example:

```
as-describe-auto-scaling-groups my-test-asg --headers
```

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

AUTO-SCALING-GROUP	GROUP-NAME	LAUNCH-CONFIG	AVAILABILITY-ZONES	MIN-SIZE	MAX-SIZE	DESIRED-CAPACITY	TERMINATION-POLICIES
AUTO-SCALING-GROUP	my-test-asg	my-test-lc	us-west-2c	1	3	2	Default
INSTANCE	INSTANCE-ID	AVAILABILITY-ZONE	STATE	STATUS	LAUNCH-CONFIG		
INSTANCE	i-98e204e8	us-west-2c	InService	Healthy	my-test-lc		
INSTANCE	i-d998ded1	us-west-2c	InService	Healthy	my-test-lc		

Check the **Instance ID** of the new instance. It matches the instance ID mentioned in the **EC2Instanceid:** field in the notification email.

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. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the Amazon EC2 **Resources** page, in the **EC2 Dashboard** pane, under **Auto Scaling**, click **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group (*my-test-asg*) from the list.
4. The bottom pane displays the details of your Auto Scaling group. In the bottom pane, click **Notifications** tab.
5. In the notifications pane, click **Delete**.

To delete Auto Scaling notification configuration using the CLI

1. Use `as-delete-notification-configuration` command by specifying the following values:
 - Auto Scaling group name: *my-test-asg*
 - ARN: *ARN-placeholder*

Note

Replace the *ARN-placeholder* with the ARN of your Amazon SNS topic that you associated with the Auto Scaling group.

Your command should look similar to the following example:

```
as-delete-notification-configuration my-test-asg --topic-ARN ARN-placeholder
```

2. After confirming that you want to delete the notification configuration, Auto Scaling returns the following:

```
OK-Deleted Notification Configuration
```

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. The error messages can be retrieved from the description of the Auto Scaling activities. You can use either the Query API or the command line interface (CLI) to retrieve an error message.

Retrieving an Error Message

To retrieve an error message from the description of Auto Scaling activities, use the `as-describe-scaling-activities` command. Alternatively, you can use the `DescribeScalingActivities` API action. For more information about the API action, see [DescribeScalingActivities](#) in the *Auto Scaling API Reference*.

The `as-describe-scaling-activities` command takes the following arguments:

```
as-describe-scaling-activities [ActivityIds [ ActivityIds... ] ] [--auto-scaling-group value][--max-records value][General Options]
```

In this example, you'll get the XML description of the Auto Scaling activities for the `MyASGroup` Auto Scaling group.

```
PROMPT>as-describe-scaling-activities --auto-scaling-group MyASGroup --show-xml
```

Auto Scaling returns the following:

```
<DescribeScalingActivitiesResponse xmlns="http://ec2.amazonaws.com/doc/2011-01-01/">
  <DescribeScalingActivitiesResult>
    <Activities>
      <member>
        <StatusCode>Failed</StatusCode>
        <Progress>0</Progress>
        <ActivityId>063308ae-aa22-4a9b-94f4-9fae70b82ad0</ActivityId>
        <StartTime>2012-04-12T17:32:07.882Z</StartTime>
        <AutoScalingGroupName>MyASGroup</AutoScalingGroupName>
        <Cause>At 2012-04-12T17:31:30Z a user request created an AutoScalingGroup
        changing the desired capacity from 0 to 1. At 2012-04-12T17:32:07Z an instance
```

```

was started in response to a difference between desired and actual capacity,
increasing the capacity from 0 to 1.</Cause>
  <Details>{}</Details>
  <Description>Launching a new EC2 instance. Status Reason: The image id
'ami-4edb0327' does not exist. Launching EC2 instance failed.</Description>
  <EndTime>2012-04-12T17:32:08Z</EndTime>
  <StatusMessage>The image id 'ami-4edb0327' does not exist. Launching EC2
instance failed.</StatusMessage>
</member>
</Activities>
</DescribeScalingActivitiesResult>
<ResponseMetadata>
  <RequestId>7a641adc-84c5-11e1-a8a5-217eb05262e2</RequestId>
</ResponseMetadata>
</DescribeScalingActivitiesResponse>

```

The response includes a list of `Activities` associated with the `MyASGroup` Auto Scaling group. The `StatusCode` contains the current status of the activity. The `StatusMessage` contains the error message, which is the verbose description of the activity status.

Troubleshooting Auto Scaling issues involves looking at how your Amazon EC2 AMIs and instances are configured. You can create, access, and manage your AMIs and instances using one of the Amazon EC2 interfaces: the AWS Management Console, the command line interface (CLI), or the Query API. You must install the Amazon EC2 command line tools before you can use them. For more information see [Getting Started with the Command Line Tools](#) in the *Amazon EC2 User Guide for Linux Instances*. For information about using the Query API, see [Making API Requests](#) in the *Amazon EC2 User Guide for Linux Instances*.

The following tables list the types of error messages and provide links to the troubleshooting resources that you can use as you work with your Auto Scaling issues.

Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure

Issue	Error Message
Auto Scaling group	AutoScalingGroup <Auto Scaling group name> not found. (p. 262)
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request (p. 262)
AWS account	You are not subscribed to this service. Please see http://aws.amazon.com. (p. 262)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 262)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 263)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 263)
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>).... (p. 262)
Key pair	The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 261)
Launch configuration	The requested configuration is currently not supported. (p. 261)

Issue	Error Message
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 263)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 261)

Troubleshooting Auto Scaling: Amazon EC2 AMIs

Issue	Error Message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 264)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 264)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 265)
Architecture mismatch	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. 265)
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. 264)

Troubleshooting Auto Scaling: Load Balancer Configuration

Issue	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 265)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 266)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 266)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 266)

Troubleshooting Auto Scaling: Capacity Limits

Issue	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 267)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)... (p. 267)

Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure

The following topics provide information about your EC2 instances that fail to launch, potential causes, and the steps you can take to resolve the issues.

When your EC2 instances fail to launch, you might get one or more of the error messages covered in the following topics. To retrieve an error message and to review the error message lists sorted by the type of issue, see [Retrieving an Error Message](#) (p. 258).

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 [DescribeSecurityGroups](#) action or `ec2-describe-group` command to get the list of the security groups associated with your account.
 2. From the list, select the security groups you want to use. To create a new security group use the [CreateSecurityGroup](#) action or the `ec2-create-group` command.
 3. Create a new launch configuration.
 4. Update your Auto Scaling group with the new launch configuration using the [UpdateAutoScalingGroup](#) action or the `as-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 [DescribeKeyPairs](#) action or the `ec2-describe-keypairs` command to get the list of the key pairs available to you.
 2. From the list, select the key pairs you want to use. To create a new key pair, use [CreateKeyPair](#) action or `ec2-create-keypair` command.
 3. Create a new launch configuration.
 4. Update your Auto Scaling group with the new launch configuration using the [UpdateAutoScalingGroup](#) action or the `as-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 [UpdateAutoScalingGroup](#) action or the `as-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 AWS Management Console, the [DescribeVolumes](#) action, or the `ec2-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 [UpdateAutoScalingGroup](#) action or the `as-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 value of the `virtualName` parameter in the format: `ephemeral<number>`. 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 [UpdateAutoScalingGroup](#) action or the `as-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 about block device mapping, 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 [UpdateAutoScalingGroup](#) action or the `as-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 cluster placement groups, see [Using Cluster Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
 2. Follow the instructions detailed in the [Creating a Cluster Placement Group](#) section of the *Using Cluster Instances* topic 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 the new launch configuration using the [UpdateAutoScalingGroup](#) action or the `as-update-auto-scaling-group` command.

Troubleshooting Auto Scaling: Amazon EC2 AMIs

The following topics provide information about the issues associated with your Amazon EC2 AMIs, potential causes, and the steps you can take to resolve the issues.

When your EC2 instances fail to launch due to issues with your AMIs, you might get one or more of the error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see [Retrieving an Error Message \(p. 258\)](#).

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 [UpdateAutoScalingGroup](#) action or the `as-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/UNIX 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 [UpdateAutoScalingGroup](#) action or the `as-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:**
 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 [UpdateAutoScalingGroup](#) action or the `as-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 [UpdateAutoScalingGroup](#) action or the `as-update-auto-scaling-group` command.

Troubleshooting Auto Scaling: Load Balancer Configuration

The following topics provide 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.

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 error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see [Retrieving an Error Message](#) (p. 258).

Before you begin troubleshooting issues with the load balancer configurations, be sure you've installed the Elastic Load Balancing interface you plan to use to access your load balancer. For more information, see [Get Set Up With Elastic Load Balancing Interfaces](#) in the *Elastic Load Balancing Developer Guide*.

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 either the [DescribeLoadBalancer](#) action or the `elb-describe-lbs` 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.

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

The following topics provide information about issues with the capacity limits of your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

When 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 error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see [Retrieving an Error Message](#) (p. 258).

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 [UpdateAutoScalingGroup](#) action or the `as-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 `DesiredCapacity` parameter using the [UpdateAutoScalingGroup](#) action or the `as-update-auto-scaling-group` command.
 - If you've reached the limit for number of EC2 instances, see [Contact Us](#) and place a request to raise your Amazon EC2 instance limit.

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 the important changes to the *Auto Scaling Developer Guide*.

- API version: 2011-01-01
- Latest documentation update: October 1, 2014

Change	Description	Release Date
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 Add, Modify, or Remove Auto Scaling Group Tags (p. 158) .	01 May 2014
Support for launching dedicated instances in a VPC	You can now launch Dedicated Instances in a VPC by specifying a placement tenancy attribute when you create a launch configuration. For more information, see Choosing Your Instance Placement Tenancy (p. 129) .	23 April 2014
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 Create a Launch Configuration Using an EC2 Instance (p. 111) For information about creating an Auto Scaling group using an EC2 instance, see Create an Auto Scaling Group Using an EC2 Instance ID (p. 122) .	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 Attach EC2 Instances to Your Auto Scaling Group (p. 202) .	02 January 2014
Account limits	You can now find out the number of Auto Scaling resources allowed for your account by using either an Auto Scaling CLI or query action. For more information, see Auto Scaling Limits (p. 7) .	02 January 2014

Change	Description	Release Date
Support for Auto Scaling in the AWS Management Console	You can now access Auto Scaling using the AWS Management Console. For more information , see Getting Started with Auto Scaling (p. 28) .	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 Auto Scaling in Amazon Virtual Private Cloud (p. 128) .	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 Configuring Your Auto Scaling Groups (p. 184) .	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 Scaling Based on Amazon SQS (p. 88) .	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 Scaling the Size of Your Auto Scaling Group (p. 45) .	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 Choosing a Termination Policy (p. 49) .	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 Launch Auto Scaling Instances with an IAM Role (p. 108) .	11 June 2012
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 Launching Spot Instances in Your Auto Scaling Group (p. 162) .	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 Troubleshooting Auto Scaling (p. 258) .	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 Add, Modify, or Remove Auto Scaling Group Tags (p. 158) .	26 January 2012

Change	Description	Release Date
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 Getting Notifications When Your Auto Scaling Group Changes (p. 248).</p> <p>Auto Scaling also added the following new features:</p> <ul style="list-style-type: none"> • The ability to set up recurring scaling activities using cron syntax. For more information, see the PutScheduledUpdateGroupAction API command. • A new configuration setting that allows you to scale out without adding the launched instance to the load balancer (LoadBalancer). For more information, see the ProcessType API data type. • The <code>ForceDelete</code> flag in the <code>DeleteAutoScalingGroup</code> command that tells the service to delete the Auto Scaling group with the instances associated to it without waiting for the instances to be terminated first. For more information, see the DeleteAutoScalingGroup API command. 	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 Load Balance Your Auto Scaling Group (p. 185).	12 January 2011
Scheduled scaling actions	Added the ability to create scheduled scaling actions. For more information, see Scheduled Scaling (p. 78).	2 December 2010
Amazon VPC support	Added support for Amazon VPC. For more information, see Auto Scaling in Amazon Virtual Private Cloud (p. 128).	2 December 2010
HPC clusters	Added support for high performance computing (HPC) clusters.	2 December 2010
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 Add an Elastic Load Balancing Health Check to your Auto Scaling Group (p. 195).	2 December 2010
CloudWatch alarms	Removed the older Trigger mechanism and redesigned Auto Scaling to use the CloudWatch alarm feature. For more information, see Dynamic Scaling (p. 57).	2 December 2010
Suspend and resume scaling	Added a new feature that lets you suspend and resume scaling processes.	2 December 2010
IAM support	This service now integrates with IAM. For more information, see Controlling Access to Your Auto Scaling Resources (p. 105).	2 December 2010

Change	Description	Release Date
New Region	Auto Scaling now supports the Asia Pacific (Singapore) region. For more information, see Regions and Endpoints .	28 April 2010

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.