Auto Scaling

Developer Guide

API Version 2011-01-01

Amazon Web Services

Auto Scaling: Developer Guide

Amazon Web Services

Welcome ... 1
Understanding Auto Scaling ... 2
Auto Scaling Concepts ... 3
Types of Scaling ... 9
Auto Scaling Tools ... 10
Configuring Auto Scaling ... 21
Auto Scaling and AWS Identity and Access Management ... 28
Using Auto Scaling with AWS Products ... 30
Using Auto Scaling ... 36
Basic Scenario in Auto Scaling ... 36
Auto Scaling With Email Notifications ... 40
Auto Scaling with Alarms and Load Balancing ... 45
Using Auto Scaling to Launch Spot Instances ... 50
Launching Auto Scaling Instances with an IAM Role ... 61
Expand to an Additional Availability Zone ... 65
Merge into a Single Multi-Zone Group ... 66
Shut Down an Auto-Scaled, Load-Balanced Application ... 69
Suspend and Resume Processes ... 73
Tagging Auto Scaling Groups and Amazon EC2 Instances ... 74
Troubleshooting Auto Scaling ... 83
Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure ... 86
Troubleshooting Auto Scaling: Amazon EC2 AMIs ... 89
Troubleshooting Auto Scaling: Load Balancer Configuration ... 90
Troubleshooting Auto Scaling: Capacity Limits ... 91
Document History ... 95
Glossary ... 93
Index ... 98

Welcome

This is the *Auto Scaling Developer Guide*. This guide contains conceptual information about the Auto Scaling web service, as well as information about how to use the service to create new web applications or integrate with existing ones. Separate sections describe how to integrate Auto Scaling with other AWS products, such as Amazon Elastic Compute Cloud, Amazon Elastic Load Balancing, Amazon Simple Notification Service, and Amazon CloudWatch.

Auto Scaling is a web service that enables you to automatically launch or terminate Amazon Elastic Compute Cloud (Amazon EC2) instances based on user-defined policies, health status checks, and schedules. This service is used in conjunction with Amazon CloudWatch and Elastic Load Balancing services.

How Do I...?

See the following table for links to information on how to work with Auto Scaling.

How Do I...	Relevant Resources
Learn more about the business case for Auto Scaling	Auto Scaling product information
Learn how Auto Scaling works	Understanding Auto Scaling (p. 2)
Get started with Auto Scaling	Get Started with Auto Scaling
Decide whether Auto Scaling is the right choice for my use case	Using Auto Scaling (p. 36)
Configure Auto Scaling	Configuring Auto Scaling
Control and manage user identity and access	Auto Scaling and AWS Identity and Access Management (p. 28)
Get the technical FAQ	Auto Scaling Technical FAQ
Get help from the community of developers	Amazon EC2 Discussion Forums

Understanding Auto Scaling

Topics

- Features (p. 2)
- Auto Scaling Concepts (p. 3)
- Types of Scaling (p. 9)
- Auto Scaling Tools (p. 10)
- Configuring Auto Scaling (p. 21)
- Auto Scaling and AWS Identity and Access Management (p. 28)
- Using Auto Scaling with AWS Products (p. 30)

Auto Scaling is a web service designed to launch or terminate EC2 instances automatically based on user-defined policies, schedules, and health checks. Auto Scaling is useful for maintaining a fleet of Amazon EC2 instances that can handle the presented load.

As its name implies, Auto Scaling responds automatically to changing conditions. All you need to do is specify how it should respond to those changes. For example, you can instruct Auto Scaling to launch an additional instance whenever CPU usage exceeds 90 percent for ten minutes, or to terminate half of your web site’s instances over the weekend when traffic is expected to be low.

You can also use Auto Scaling to ensure that the instances in your fleet are performing optimally, so that your applications continue to run efficiently. Auto Scaling groups can even work across multiple Availability Zones distinct physical locations for the hosted EC2 instances so that if an Availability Zone becomes unavailable, Auto Scaling will automatically redistribute applications to a different Availability Zone.

Features

Auto Scaling offers several features that help you save both time and money.

Elastic Capacity

Automatically add compute capacity when application usage rises and remove it when usage drops.

Ease of Use

Manage your instances spread across either one or several Availability Zones as a single collective entity, using simple command line tools or programmatically via an easy-to-use web service API.

Cost Savings

Save compute costs by terminating underused instances automatically and launching new instances when you need them, without the need for manual intervention.

Geographic Redundancy and Scalability

Distribute, scale, and balance applications automatically over multiple Availability Zones within a region.

Easier Maintenance

Replace lost or unhealthy instances automatically based on predefined alarms and thresholds.

Scheduled Actions

Schedule scaling actions for future times and dates when you expect to need more or less capacity.

Auto Scaling Concepts

Topics

- Auto Scaling Functional Overview (p. 3)
- Auto Scaling Group (p. 5)
- Health Check (p. 5)
- Launch Configuration (p. 5)
- Tagging (p. 5)
- Trigger (p. 5)
- Policy (p. 6)
- Alarm (p. 6)
- Scheduled Update (p. 6)
- Scaling Activity (p. 6)
- Suspendable Processes (p. 7)
- Availability Zones and Regions (p. 8)

This section introduces you to Auto Scaling terminology and concepts. Many of the concepts introduced in this section are discussed in more detail in later sections. The concepts are briefly presented here to give you a basic understanding of common Auto Scaling terms.

Auto Scaling Functional Overview

Auto Scaling is designed to help make using the Amazon Elastic Compute Cloud (EC2) easier by helping reduce the operational burden of deploying and maintaining applications.

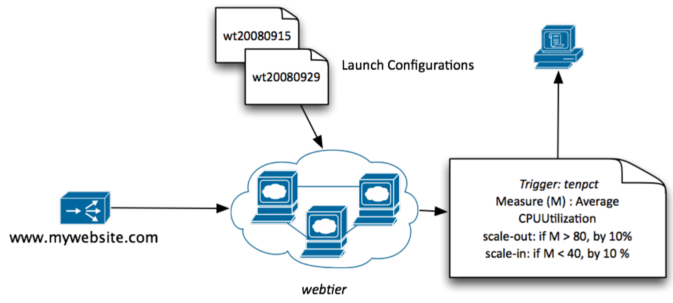
Auto Scaling monitors the health of each EC2 instance that it launches. If any instance terminates unexpectedly, Auto Scaling detects the termination and launches a replacement instance. This capability helps you maintain a fixed, desired number of EC2 instances automatically.

Auto Scaling lets you automatically adjust the size of a fleet of EC2 instances. Auto Scaling can add or remove EC2 instances from the fleet to help you seamlessly deal with load changes to your application, and let you introduce your own plans so you can proactively set the size of your Auto Scaling group.

In a common EC2 scenario, multiple copies of an application run simultaneously to cover the volume of customer traffic. Customers see only one URL for the application, but behind that simple URL are many identical EC2 instances, each handling customer requests.

These EC2 instances are categorized into Auto Scaling groups. This is the core concept of the service. Auto Scaling groups are defined with a minimum and maximum number of EC2 instances. The Auto Scaling service launches more instances (up to the defined maximum) for the Auto Scaling group to handle an increase in traffic and, as demand decreases, takes instances out of service to more efficiently use computing resources.

In the following illustration, Internet traffic is routed from the public URL into an Auto Scaling group named *webtier*. The Auto Scaling group has *triggers* that increase or decrease the size of the Auto Scaling group based on the average CPU utilization for the whole group. When a trigger fires, Auto Scaling uses a launch configuration to create a new instance.



Every Auto Scaling group you create has a launch configuration. Launch configurations enable you to describe each instance that Auto Scaling will create for you when a trigger fires. They are, essentially, a set of parameters for an EC2 `runInstances` call.

Note

The maximum number of launch configurations per account is 100.

An Auto Scaling group can have only one launch configuration at a time. However, you can modify the launch configuration, and Auto Scaling will use the modified settings to launch any new instances.

Note

If you modify the launch configuration, Auto Scaling will not apply your new settings to existing instances.

If Auto Scaling needs to scale down and remove instances, it terminates instances with old launch configurations first.

As mentioned, Auto Scaling uses triggers to indicate when to launch new instances and when to take them out of service. A trigger is a mechanism that you set to tell the system when you want to increase or decrease the number of instances. You can set a trigger to activate on any metric published to Amazon CloudWatch, such as *`CPUUtilization`*. When activated, the trigger launches a long-running process called a Scaling Activity (p. 6).

Auto Scaling supports, but does not require, AWS Elastic Load Balancing. You can configure your Auto Scaling group so that user requests are distributed across a group of EC2 instances. You can add an Elastic Load Balancing LoadBalancer to your Auto Scaling group and use Elastic Load Balancing metrics (such as request latency or request count) to scale your application.

For more information about how to use Auto Scaling APIs to set up your Auto Scaling groups, launch configurations, and triggers, see Using Auto Scaling (p. 36), and the Auto Scaling API Reference.

Auto Scaling Group

An *Auto Scaling group* is a representation of multiple Amazon 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 automatically scale the number of instances or maintain a fixed number of instances. An Auto Scaling group can contain EC2 instances that come from one or more EC2 Availability Zones.

For more information, see CreateAutoScalingGroup in the *Auto Scaling API Reference*.

Health Check

A *health check* is a call to check on the health status of each instance in an Auto Scaling group. If an instance reports degraded performance, Auto Scaling terminates the instance and launches another one to take its place. This ensures that your Auto Scaling group is consistent and operating normally. For more information, see Maintaining Current Scaling Level (p. 21).

Launch Configuration

A *launch configuration* captures the parameters necessary to create new EC2 instances. You can attach only one launch configuration to an Auto Scaling group at a time. When you attach a new or updated launch configuration to your Auto Scaling group, any new instances will be launched using the new configuration parameters. Existing instances are not affected. When Auto Scaling needs to scale down, it first terminates instances that have an older launch configuration.

For more information, go to CreateLaunchConfiguration in the *Auto Scaling API Reference*.

Tagging

An Auto Scaling group *tag* is a tool for organizing your Auto Scaling resources and providing additional information for your Auto Scaling group such as software version, role, or location. Auto Scaling group tags work like Amazon EC2 tags; Auto Scaling group tags provide search, group, and filter functionality. These tags have a key and value that you can modify. You can also remove Auto Scaling group tags any time.

For more information about using tags with Auto Scaling groups, go to Tagging Auto Scaling Groups and Amazon EC2 Instances (p. 74).

Trigger

A *trigger* is a concept that combines two AWS features: a CloudWatch alarm (configured to watch a specified CloudWatch metric) and an Auto Scaling policy that describes what should happen when the alarm threshold is crossed.

In most cases, you will need two triggers one trigger for scaling up and another for scaling down. For example, if you want to scale up when your CPU usage increases to 80 percent, you need to configure a CloudWatch alarm and create an Auto Scaling policy. The alarm detects when the CPU usage has reached 80 percent and sends a message to Auto Scaling. Auto Scaling determines what to do by using the instructions in the scaling policy. If you also want to scale down when your CPU usage decreases to 40 percent, you need a second trigger. In other words, you need to configure a separate CloudWatch alarm to detect the 40 percent threshold and create a separate Auto Scaling policy that scales down.

For more information on alarms, see Alarm (p. 6).

Policy

A *policy* is a set of instructions for Auto Scaling that tells the service how to respond to CloudWatch alarm messages. You can configure a CloudWatch alarm to send a message to Auto Scaling whenever a specific metric has reached a triggering value. When the alarm sends the message, Auto Scaling executes the associated policy on an Auto Scaling group to scale the group up or down.

For more information about alarms, go to the *Amazon CloudWatch Developer Guide*. For more information about policies, go to PutScalingPolicy in the *Auto Scaling API Reference*.

Alarm

An Amazon CloudWatch *alarm* is an object that watches over a single metric. An alarm can change state depending on the value of the metric. When an alarm changes state it executes one or more actions. To create an alarm, use the Amazon CloudWatch `PutMetricAlarm` action to specify the metric to watch, the threshold values for the metric, the number of evaluation periods, and, optionally, one or more Amazon Simple Notification Service actions to perform when the alarm changes state.

An alarm has three possible states:

- `OK` The metric is within the defined threshold
- `ALARM` The metric is outside of the defined threshold
- `INSUFFICIENT_DATA` The metric is not available, or there is not enough data available for the metric to set the alarm state to OK

An action is invoked when an alarm changes state and remains in that state for a number of time periods. This change could be from `OK` to `ALARM`, from `ALARM` to `INSUFFICIENT_DATA`, and so on. The state change has to be maintained for the number of time periods you specify.

Together with Auto Scaling policies, alarms can be configured to perform an Auto Scaling action when a CloudWatch metric has reached a predefined threshold. This functionality was previously called a "trigger."

For more information about alarms, go to the Amazon CloudWatch Developer Guide.

Scheduled Update

A *scheduled update* is a call to Auto Scaling that is scheduled for a future time. Currently, updates are supported only to min-, max-, and desired capacity. For more information about the supporting API action, go to PutScheduledUpdateGroupAction in the *Auto Scaling API Reference*.

Scaling Activity

A *scaling activity* is a long-running process that implements a change to your Auto Scaling group, such as changing the size of the group. It can also be a process to replace an instance, or to perform any other long-running operations supported by the service.

For more information about the supporting API action, go to DescribeScalingActivities in the *Auto Scaling API Reference*.

Auto Scaling Instance Termination

Auto Scaling terminates Amazon EC2 instances both in response to specific calls to the TerminateInstanceInAutoScalingGroup action, and also in response to other scaling activities. For example, Auto Scaling might terminate an instance when it rebalances an Availability Zone or when it downscales the size of your Auto Scaling group.

Auto Scaling selects an instance from the Auto Scaling group for termination, subject to the following conditions:

- Auto Scaling attempts to preserve instances with the latest launch configuration. In other words, if the Auto Scaling group contains EC2 instances with different launch configurations, Auto Scaling terminates the instance with a launch configuration that is currently not associated with the Auto Scaling group. If there are several instances with launch configurations that are not currently associated with the Auto Scaling group, Auto Scaling terminates the instance or instances running for the longest portion of a billing hour (without running over).
- Auto Scaling might terminate an instance from a specific Availability Zone to maintain balance across the zones.

After Auto Scaling determines which specific instance to terminate, it checks to see whether the instance is part of an Elastic Load Balancing group. If so, Auto Scaling instructs the load balancer to remove the instance from the load balancing group and waits for the removal to complete. If Auto Scaling determines that the instance is not part of an Elastic Load Balancing group, Auto Scaling attempts to terminate the instance by running system shutdown scripts.

Cooldown

Cooldown is the period of time after Auto Scaling initiates a scaling activity during which no other scaling activity can take place. A cooldown period allows the effect of a scaling activity to become visible in the metrics that originally triggered the activity. This period is configurable, and gives the system time to perform and adjust to any new scaling activities (such as scale-in and scale-out) that affect capacity.

If you want to initiate a scaling activity that ignores cooldown, you can use SetDesiredCapacity. The `SetDesiredCapacity` action ignores cooldown by default.

There are two common use cases for `SetDesiredCapacity`: one for users of the Auto Scaling triggering system, and another for developers who write their own triggering systems. Both use cases relate to the concept of cooldown.

In the first case, if you use the Auto Scaling triggering system, `SetDesiredCapacity` changes the size of your Auto Scaling group without regard to the cooldown period. This could be useful, for example, if Auto Scaling did something unexpected for some reason. If your cooldown period is 10 minutes, Auto Scaling would normally reject requests to change the size of the group for that entire 10 minute period. The `SetDesiredCapacity` command allows you to circumvent this restriction and change the size of the group before the end of the cooldown period.

In the second case, if you write your own triggering system, you can use `SetDesiredCapacity` to control the size of your Auto Scaling group. If you want the same cooldown functionality that Auto Scaling offers, you can configure `SetDesiredCapacity` to honor cooldown by setting the `HonorCooldown` parameter to `true`.

Suspendable Processes

You might want to stop automated scaling processes on your groups to perform manual operations or to turn off the automation in emergency situations. To meet these needs Auto Scaling offers two actions: SuspendProcesses and ResumeProcesses. You can suspend scaling processes at any time. When you're ready, you can resume any or all of the suspended processes.

If you suspend all of an Auto Scaling group's scaling processes, Auto Scaling creates no new scaling activities for that group for any reason. Scaling activities that were already in progress before the group was suspended continue until complete. Changes made to the group by calls to SetDesiredCapacity and UpdateAutoScalingGroup still take effect immediately. However, Auto Scaling will not create new scaling activities when there's a difference between the desired size and the actual number of instances.

You can suspend one or more of the following Auto Scaling process types by specifying them in a call to `SuspendProcesses`.

If you suspend...	Auto Scaling...
AlarmNotification	Ignores all Amazon CloudWatch notifications.
AZRebalance	Does not attempt active rebalancing. If, however, Auto Scaling initiates the launch or terminate processes for other reasons, Auto Scaling will still launch new instances in underpopulated Availability Zones and terminate existing instances in overpopulated Availability Zones.
HealthCheck	Will not automatically check instance health. Auto Scaling will still replace instances that you have marked as unhealthy with the SetInstanceHealth API call.
Launch	Does not launch new instances for any reason. Suspending the Launch process effectively suspends the AZRebalance and ReplaceUnhealthy processes.
ReplaceUnhealthy	Does not replace instances marked as unhealthy. Auto Scaling continues to automatically mark instances as unhealthy.
ScheduledActions	Suspends processing of scheduled actions. Auto Scaling silently discards any action scheduled to occur during the suspension.
Terminate	Does not terminate new instances for any reason. Suspending the Terminate process effectively suspends the AZRebalance and ReplaceUnhealthy processes.

You can suspend all Auto Scaling process types by calling `SuspendProcesses` without specifying any process types.

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 zero running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time in launching any instances.

Important

To resume processes, whether the suspension was manual (using `SuspendProcesses`) or administrative, use either the `ResumeProcesses` API action or the `as-resume-processes` CLI command.

Availability Zones and Regions

Amazon cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in several different physical locations. These locations are 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, go to 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.

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 attempts 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. 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 call `TerminateInstanceInAutoScalingGroup`, which causes the group to become unbalanced.
- An Availability Zone that previously had insufficient capacity recovers and has additional capacity available.

Auto Scaling always 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, 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 during a rebalancing activity (or by a 1-instance margin, whichever is greater). The margin is extended only if the group is at or near maximum capacity and needs rebalancing, either as a result 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.

Types of Scaling

Auto Scaling provides three types of scaling for your system: manual, by schedule, and by policy.

Manual Scaling

Manual scaling is the most basic way to scale your resources. Send an API call or use the Auto Scaling command line interface (CLI) to launch or terminate an Amazon EC2 instance. You need only specify the change in capacity you want. Auto Scaling manages the process of creating or destroying instances, including all the parameters to the Amazon EC2 `runInstance` call.

Scaling by 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.

Scaling by Policy

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 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 up and one for scaling down, for each event that you want to monitor. For example, if you want to scale up when the network bandwidth reaches a certain level, you'll create a policy telling Auto Scaling to fire up a certain number of instances to help with your traffic. But you also want an accompanying policy to scale down by a certain number when the network bandwidth level goes back down.

Auto Scaling Tools

Topics

- Using the Command Line Tools (p. 10)
- Using the Query API (p. 17)

Currently, there are two tools you can use to work with Auto Scaling. You can use the command line tools (CLI), or you can use the Query API. The command line tools must be installed on the computer with which you access your Amazon EC2 resources on the AWS cloud. This section discusses how you can use these tools.

Using the Command Line Tools

Topics

- Setting the Java Home Variable (p. 10)
- Setting Up the Tools (p. 11)
- Using Credentials (p. 15)

This section describes how to set up your environment for use with the Auto Scaling command line tools.

An installation of a Java 5–compatible Java Runtime Environment (JRE) is required. Additionally, accessing Linux and UNIX instances requires access to an SSH client and accessing Windows instances requires access to a Remote Desktop client. For more information, refer to the two following sections.

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 tools require Java version 5 or later to run. Either a JRE or JDK installation is acceptable. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to http://java.sun.com/j2se/1.5.0/.

The command line tools depend on an environment variable (`JAVA_HOME`) to locate the Java runtime. This environment variable should be set to the full path of the directory that contains a subdirectory named `bin` that in turn contains the `java` (on Linux and UNIX) or the `java.exe` (on Windows) executable. You might want to simplify the process by adding this directory to your path before other versions of Java. Make sure you don't include the bin directory in the path; that's a common mistake some users make. The command line tools won't work if you do.

Note

If you are using Cygwin, AWS_AUTO_SCALING_HOME, EC2_PRIVATE_KEY, and EC2_CERT, you must use Linux/UNIX paths (e.g., /usr/bin instead of C:\usr\bin). However, JAVA_HOME should have a Windows path. Additionally, the value of AWS_AUTO_SCALING_HOME cannot contain any spaces, even if the value is quoted or the spaces are escaped.

The following is an example of how to set this environment variable in Linux and UNIX.

```
$ export JAVA_HOME=<PATH>
```

The following is an example of the syntax in Windows.

```
C:\> set JAVA_HOME=<PATH>
```

You can confirm this by running **`$JAVA_HOME/bin/java -version`** and checking the output.

```
$ $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 Tools

Topics

- How to Get the Command Line Tool (p. 11)
- How to Tell the Tools Where They Live (p. 12)
- How to Tell the Tools Who You Are (p. 12)
- How to Change the Region (p. 13)

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

How to Get the Command Line Tool

The command line tool is available as a ZIP file in the Auto Scaling Command Line Tools. These tools are written in Java and include shell scripts for both Windows 2000/XP 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 tools to use your AWS account credentials. These are discussed next.

How to Tell the Tools Where They Live

The command line tools depend on an environment variable (`AWS_AUTO_SCALING_HOME`) to locate supporting libraries. You'll need to set this environment variable before you can use the tools. You should set this variable to the path of the directory into which the command line tools were 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`.

On Linux and UNIX, you can set this environment variable as follows:

```
$ export AWS_AUTO_SCALING_HOME=<path-to-tools>
```

On Windows the syntax is slightly different:

```
C:\> set AWS_AUTO_SCALING_HOME=<path-to-tools>
```

In addition, to make your life a little easier, you probably want to add the tools' `bin` directory to your system `PATH`. The rest of this guide assumes that you've done this.

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
```

Note

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

How to Tell the Tools Who You Are

You must also provide your AWS credentials to the command line tools. You can use your AWS access keys or your AWS X.509 certificates.

To use access keys with the command line tools

1. Log in to the AWS security credentials web site.
2. Retrieve an access key and its corresponding secret key. For instructions on how to get these keys, see How to Get Your Access Key ID and Secret Access Key (p. 16).
3. Open the file `$AWS_AUTO_SCALING_HOME/credential-file-path.template` (`%AWS_AUTO_SCALING_HOME%\credential-file-path.template` on Windows) that you downloaded as part of the command line tools ZIP file. Add your access key and secret key to the appropriate locations in the template file. Save the file to a convenient location on your workstation. You should use a new name for the file and, on Linux, set its file permissions using `chmod 600` **`file name`**.
4. Use this file in either of two ways:

- Set the AWS_CREDENTIAL_FILE environment variable to the fully qualified path of the file you just created.
- Specify the `--aws-credential-file` **`file name`** parameter with each command you use.

Alternatively, you can specify your access keys directly on the command line by including the `--I` *[your access key]* `--S` *[your secret key]* parameters.

Note

Many developers find that creating a credential file and a corresponding AWS_CREDENTIAL_FILE environment variable is the most convenient way to supply credentials to the command line tools.

To use your X.509 certificate files with the command line tools

1. Log in to the AWS security credentials site.
2. Click the **X.509 Certificate** tab, and follow the instructions to download your certificate and private key files to a secure location on your workstation. Name the files appropriately (for example, my-aws-cert.pem and my-aws-pk.pem).
3. Use these files in either of two ways:
   - Specify the `--ec2-cert-file-path=` **`certificate file name`** and `--ec2-private-key-file-path` **`key file name`** parameters with each command you use.
   - Set the EC2_CERT environment variable to the fully qualified path of the certificate file you just created, and set the EC2_PRIVATE_KEY environment variable to the fully qualified path of the key file you just created. This method saves you the effort of specifying two parameters with each command you use.

How to Change the Region

By default, the Auto Scaling tools use the US East (Northern Virginia) Region (us-east-1) with the `autoscaling.us-east-1.amazonaws.com` service endpoint URL.

If you want to explicitly specify the Region for your Auto Scaling service, set AWS_AUTO_SCALING_REGION to the service endpoint URL (`autoscaling.us-east-1.amazonaws.com`, for example).

To specify a different Region

1. View available Regions by going to Regions and Endpoints.
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`*.

   - For Linux and UNIX:

   ```
   $ export AWS_AUTO_SCALING_URL=https://<service_endpoint>
   ```

   - For Windows:

```
C:\> set AWS_AUTO_SCALING_URL=https://<service_endpoint>
```

You're ready to start using Auto Scaling.

Using Credentials

Topics

- How to Log In with Your Amazon Login and Password (p. 15)
- How to View Your AWS Access Credentials (p. 16)
- How to Get Your Access Key ID and Secret Access Key (p. 16)
- How to Create an X.509 Certificate and Private Key (p. 16)
- Viewing Your Account ID (p. 17)

This section describes how to use 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 How to Log In with Your Amazon Login and Password (p. 15).
- **Access Key ID and Secret Access Key** Used to make Query and REST-based requests. Also commonly used by UI-based tools, such as ElasticFox. For more information, see How to Get Your Access Key ID and Secret Access Key (p. 16).
- **X.509 Certificate and Private Key** Used by the command line tools and SOAP API. For more information, see How to Create an X.509 Certificate and Private Key (p. 16).
- **Account ID** Used to share resources with other AWS accounts. For more information, see Viewing Your Account ID (p. 17).

How to Log In with Your Amazon Login and Password

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.

This section describes how to log in with your login and password.

To log in with your login and password (if you have an existing account)

1. Go to the AWS web site.
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 the **Sign In** button.

To get a new Amazon login and password (create a new AWS account)

1. Go to the AWS web site.
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 the **Sign In** button.
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, go to http://aws.amazon.com/mfa.

How to View Your AWS Access Credentials

You can reuse active AWS access credentials that you've created in the past.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Security Credentials**.
3. Under **Your Account**, click **Security Credentials**.
4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.
5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

How to Get Your Access Key ID and Secret Access Key

The Access Key ID and Secret Access Key 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.

To get your Access Key ID and Secret Access Key

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 **Access Credentials** section and verify that the **Access Keys** tab is selected.
4. Locate an active Access Key in the **Your Access Keys** list.
5. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.
6. Write down the keys or save them.
7. If no Access Keys appear in the list, click **Create a New Access Key** and follow the on-screen prompts.

How to 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 once. If you lose it, you will need to create a new certificate. Up to two certificates can be active at any time.

This section describes how to create a new certificate.

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.

Viewing 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.

For information on how to share AMIs, see Using Shared AMIs. For information on how to share snapshots, see How to Modify Snapshot Permissions.

Note

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

Using the Query API

Topics

- Endpoints (p. 17)
- Making Query Requests (p. 17)

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

Making Query Requests

Topics

- Query Parameters (p. 18)
- The Request ID (p. 18)

- Request Authentication (p. 18)
- Query Example (p. 20)

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named *Action* or *Operation*. Action is used throughout this documentation, although Operation is supported for backwards compatibility with other AWS Query APIs.

Query Parameters

Each query request must include some common parameters to handle authentication and selection of an action. For more information, go to Common Query Parameters in the *Auto Scaling API Reference*.

Note

Some API operations take lists of parameters. These lists are specified using the following notation: param.member.*n*. Values of *n* are integers starting from 1. All lists of parameters must follow this notation, including lists that contain only one parameter. For example, a query parameter list looks like this:

```
&attribute.member.1=this
&attribute.member.2=that
```

The Request ID

In every response from AWS, you will see the element `ResponseMetadata`, which contains a string element called `RequestId`. This is simply a unique identifier that AWS assigns to this request for tracking and troubleshooting purposes.

Request Authentication

You can send Query requests over either HTTP or HTTPS. Regardless of which protocol you use, you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*.

To create the signature

1. Create the canonicalized query string that you need later in this procedure:

   a. Sort the UTF-8 query string components by parameter name with natural byte ordering.
   The parameters can come from the GET URI or from the POST body (when `Content-Type` is `application/x-www-form-urlencoded`).

   b. URL-encode the parameter name and values according to the following rules:

      - Do not URL-encode any of the unreserved characters that RFC 3986 defines.
        These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
      - Percent-encode all other characters with `%XY`, where X and Y are hex characters 0-9 and uppercase A-F.
      - Percent-encode extended UTF-8 characters in the form `%XY%ZA`, and so on.
      - Percent-encode the space character as `%20` (and not +, as common encoding schemes do).

Note

Currently, all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the `"\n"` represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
               ValueOfHostHeaderInLowercase + "\n" +
               HTTPRequestURI + "\n" +
               CanonicalizedQueryString <from the preceding step>
```

   The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to but not including the query string. If the `HTTPRequestURI` is empty, use a forward slash (/).
3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.
   For more information, go to http://www.ietf.org/rfc/rfc2104.txt.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the *`Signature`* request parameter.

Important

The final signature you send in the request must be URL-encoded as specified in RFC 3986 (for more information, go to http://www.ietf.org/rfc/rfc3986.txt). If your toolkit URL-encodes your final request, then it handles the required URL-encoding of the signature. If your toolkit doesn't URL-encode the final request, then make sure to URL-encode the signature before you include it in the request. Most importantly, make sure the signature is URL-encoded *only once*. A common mistake is to URL-encode it manually during signature formation, and then again when the toolkit URL-encodes the entire request.

Query Example

Example Describe AutoScalingGroup API Request

This example uses `CreateAutoScalingGroup`.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=webtier
&LaunchConfigurationName=wt20080929
&MinSize=0
&MaxSize=2
&DefaultCooldown=0
&Expires=2011-02-10T12%3A00%3A00Z
&AvailabilityZones.member.1=us-east-1c
&Action=CreateAutoScalingGroup
&Version=2011-01-01
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
```

The following is the string to sign.

```
GET\n
autoscaling.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=CreateAutoScalingGroup
&AutoScalingGroupName=webtier
&AvailabilityZones.member.1=us-east-1c
&DefaultCooldown=0
&Expires=2011-02-10T12%3A00%3A00Z
&LaunchConfigurationName=wt20080929
&MinSize=0
&MaxSize=2
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Version=2011-01-01
```

The following is the signed request.

```
http://autoscaling.amazonaws.com/?AutoScalingGroupName=webtier
&LaunchConfigurationName=wt20080929
&MinSize=0
&MaxSize=2
&DefaultCooldown=0
&AvailabilityZones.member.1=us-east-1c
&Action=CreateAutoScalingGroup
&Version=2011-01-01
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
&Signature=<URLEncode(Base64Encode(Signature))>
&Expires=2011-02-10T12%3A00%3A00Z
```

Configuring Auto Scaling

Topics

- Maintaining Current Scaling Level (p. 21)
- Scaling by Policy (p. 23)
- Scaling by Schedule (p. 26)

When you configure Auto Scaling, you tell the service how to do one of three things:

- Maintain current instance levels (health check)
- Create more instances (scale up)
- Delete current instances (scale down)

You can maintain current instance levels based on the health status of the instances in your Auto Scaling group. You can scale up or down based on a policy or a scheduled action.

A common use of Auto Scaling is to maintain current instance levels by conducting health status checks. If an instance fails and is no longer a useful part of an application fleet, the Auto Scaling service can replace it with a new, working instance.

A scaling policy tells Auto Scaling to perform a scaling action when the value of a certain metric crosses a particular threshold. For example, you might design a policy that calls for a 10 percent increase in instances when *`CPUUtilization`* for your Auto Scaling group reaches 80 percent. You would also design another policy to scale down when *`CPUUtilization`* falls below 40 percent.

A scheduled action tells Auto Scaling to perform a scaling action at certain time. For example, if you schedule a product promotion and expect higher traffic at release time, you might schedule more instances to accommodate that traffic at the release date. You would probably schedule a corresponding scaling action to scale down instances when you expect traffic to decrease.

It is possible to combine health checks, scaling policies, and scheduled actions. For example, you might want to increase and decrease instances based on CPU utilization during your high traffic dates, but also ensure that unhealthy instances are terminated and replacements are launched.

The various types of scaling activities and health checks are discussed more thoroughly in the sections that follow.

Maintaining Current Scaling Level

One way to use Auto Scaling is to ensure that your current instances are running and in good shape. Auto Scaling provides this service by using a health check on current instances. When Auto Scaling finds that an instance is unhealthy, it terminates that instance and starts a new one.

This section provides details about health checks and how to integrate them into your Auto Scaling group.

Overview

All instances 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 three sources: Amazon EC2, Elastic Load Balancing, and the `SetHealthStatus` action in the Auto Scaling API.

All Auto Scaling groups get Amazon EC2 health checks by default. If you choose to use the Elastic Load Balancing health check, Auto Scaling still bases health decisions on the Amazon EC2 health information in addition to the Elastic Load Balancing information. This means that an Amazon EC2 instance can still be marked unhealthy and replaced even if the Elastic Load Balancing considers it healthy. This could happen if Amazon EC2 determines that the hardware for the instance has become degraded, but the application is still passing Elastic Load Balancing health checks.

Once an instance has been marked unhealthy, it is almost immediately scheduled for replacement. It will never automatically recover its health. You can intervene manually by calling the `SetInstanceHealth` action 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 `SetInstanceHealth` is probably useful only for a suspended group. For more information about suspending and resuming processes, see Suspendable Processes (p. 7).

Auto Scaling creates a new `TerminateInstance` scaling activity for an unhealthy instance and terminates it. Subsequently, a new `LaunchInstance` scaling activity brings a replacement instance into service.

Creating a Health Check for Elastic Load Balancing

Use `UpdateAutoScalingGroup` to create a health check on an existing Auto Scaling group. By default, Auto Scaling uses the Amazon EC2 health status for all Auto-Scaling-managed instances. To also use the Elastic Load Balancer's health check, set the `HealthCheckType` property of the group to `ELB`. Frequently, new instances need to warm up, briefly, before they can pass a health check. To provide ample warm-up time, set the `HealthCheckGracePeriod` property (`--grace-period` in the Auto Scaling CLI tool) of the group to match the expected startup period of your application:

```
% as-update-auto-scaling-group myGroup –-health-check-type ELB  –-grace-period
 300
```

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`.

Listing the Health Status

Use `DescribeAutoScalingGroups` to return information about the instances in your Auto Scaling group. Instances have a `HealthStatus` field that is either `Healthy` or `Unhealthy`. Unhealthy instances are short-lived and won’t remain in your Auto Scaling group very long.

Customizing Health Checks

If you have your own health check system, you can integrate it with Auto Scaling. Use `SetInstanceHealth` to send the instance's health information directly from your system to Auto Scaling. Auto Scaling enforces the health check grace period by rejecting changes in health status for new instances that are still within the grace period.

```
% as-set-instance-health i-a1b2c3 –-status Unhealthy
```

Scaling by Policy

A scaling policy tells Auto Scaling how to change the size of your application fleet in response to load changes, enabling you to specify not only whether you want to scale your group up or down, but also how much. You can express the desired change in capacity as an absolute number, an increment, or as a percentage of the current group size. When you execute a policy, Auto Scaling uses both the current group capacity and the desired change specified in the policy to compute a new desired capacity. Auto Scaling then updates the desired capacity and thus, the size of your fleet.

Each Auto Scaling group can have up to 25 policies.

Important

We recommend that you create two policies for each scaling change you want to perform. You need one policy to scale up and another policy to scale down.

Creating a Policy

Use `PutScalingPolicy` to specify a scaling adjustment and type for your group. The type specifies whether the adjustment is an absolute value, a constant increment, or a percentage of the current capacity. A positive value adds to the current capacity and a negative value removes from the current capacity. For example, if you want to scale up your group by 100 percent, set the `increment` to `100` and the `type` to `PercentChangeInCapacity`:

Note

When you are using Windows tools, you must use quotation marks.

```
# When scaling up, double my group's current capacity
%as-put-scaling-policy my-group --auto-scaling-group double-group-size
--adjustment 100 --type PercentChangeInCapacity
```

And then use the corresponding policy to scale back down:

```
# When scaling down, decrease the capacity by 1
%as-put-scaling-policy my-group --auto-scaling-group scale-down
--adjustment=-1 --type Absolute
```

A successful `PutScalingPolicy` action returns an Amazon Resource Name (ARN), which 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.

Note

All Auto Scaling names, including policy names, cannot contain the colon (:) character because colons serve as delimiters in ARNs.

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, an adequate cooldown period helps to prevent a trigger from firing based on stale metrics. To use a different cooldown period than the default specified in the Auto Scaling group, use `PutScalingPolicy`, specifying your custom cooldown period. If the policy does not specify a cooldown, the group's default cooldown is used.

```
# Use a cooldown of 10 minutes when executing policy 'double-group-size'
%as-put-scaling-policy my-group --auto-scaling-group double-group-size
--adjustment 100 --type PercentChangeInCapacity --cooldown 600
```

Executing a Policy

Use `ExecutePolicy` to specify a policy you want to run. Auto Scaling scales your Auto Scaling group according to the specified policy. You can use `ExecutePolicy` to test a policy that you will later use with a CloudWatch alarm.

```
# Scale my-group as specified by scale-up-exponential policy
      %as-execute-policy my-group --auto-scaling-group double-group-size
```

To force Auto Scaling to ignore a policy execution command while your Auto Scaling group is in cooldown, specify the `HonorCooldown` parameter when you make the `ExecutePolicy` call. Auto Scaling will reject your call with an appropriate error message if your group is in cooldown. The `HonorCooldown` flag is probably most useful for automated processes. You can use it to make sure that your custom scaling scripts respect cooldown periods.

Listing Your Policies

Use `DescribePolicies` to list the policies attached to your Auto Scaling group.

```
# List all the policies attached to group my-group
% as-describe-policies my-group
POLICY  POLICY-NAME  GROUP-NAME       ADJUSTMENT    TYPE
COOLDOWN
POLICY  my-group    double-group-size   100          PercentChangeInCapacity
POLICY  my-group    scale-down           -1          ExactCapacity
POLICY  my-group    scale-up-fast        10          ExactCapacity
POLICY  my-group    scale-up              1          ExactCapacity
```

To determine not only which policies were executed, but also when they were executed, use `DescribeScalingActivities`. The `cause` parameter lists both the specific policy executed as well as the resulting activity.

```
Description => Launching a new EC2 instance: i-abcdefg
Cause => At  2010-07-06T18:32:02Z, user executed policy 'scale-up' changed de
sired capacity from 1 to 2. At  2010-07-06T18:32:13Z, an instance was started
in response to a difference between desired and actual capacity.

Description => Terminating EC2 instance i-abcdefg
Cause => At  2010-07-06T19:44:10Z, user executed policy 'scale-down' changed
desired capacity from 2 to 1. At  2010-07-06T19:44:35Z, an instance was termin
ated in response to a difference between desired and actual capacity.
```

Updating a Policy

Use `PutScalingPolicy` with the name of the existing policy to update the scaling increment value and type. Note, however, that this action effectively deletes the previously existing policy and creates an entirely new policy with the settings you specify in the more recent call.

Deleting a Policy

Use `DeletePolicy` with the name of the existing policy to delete the policy.

Suspending a Policy

Use `SuspendProcesses` to suspend any policy executions on the specified Auto Scaling group.

```
# Suspend execution of policies
%as-suspend-processes my-group
```

To resume policy executions, use `ResumeProcesses`.

```
# Suspend execution of policies
      %as-suspend-processes my-group
```

Note

You can't suspend executions of individual policies.

Scaling by Schedule

A time-based scaling plan is a set of instructions for Auto Scaling to perform at a specific time in the future. These are called scheduled actions and you can use these actions to scale an Auto Scaling group in response to predictable load changes.

To create a time-based scaling plan, you specify the time at which you want the plan to take effect, and 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 plan.

Note

If you try to schedule your action in the past, Auto Scaling returns an error message.

Topics

- Creating Scheduled Actions (p. 26)
- Listing Scheduled Actions (p. 26)
- Updating Scheduled Actions (p. 27)
- Suspending Scheduled Actions (p. 27)
- Deleting Scheduled Actions (p. 27)
- Managing Scheduled Actions (p. 27)
- Programming Considerations (p. 27)

Creating Scheduled Actions

Use Auto Scaling's `PutScheduledUpdateGroupAction` to configure scheduled actions. This call enables you to specify the date and time for each action, along with the minimum, maximum, and desired size of the group you want to take effect at that point in time. For example, if you knew you were having a book sale on April 5th, you would scale up in time for the sale:

```
# Make sure we scale up in time for the book sale on the 5th
% as-put-scheduled-update-group-action my-group –-name “book-scale-up”
-- time “2010-04-05T02:00:00Z” –-min 500 –-max 500
```

Then you would release excess capacity after the sale:

```
# The sale is over by the 6th, so shut down excess capacity
% as-put-scheduled-update-group-action my-group –-name “book-sale-over”
-- time “2010-04-06T00:00:00Z” –-min 100 –-max 100
```

Listing Scheduled Actions

Use `DescribeScheduledActions` to list all the activity plans attached to your Auto Scaling group. The listing displays activities that are still waiting to be executed. Once a scheduled action is completed, it is automatically deleted and, thus, no longer visible in the list of planned actions. Instead, an activity record for the running activity will be displayed with `DescribeScalingActivities`.

```
% as-describe-scheduled-actions my-group
NAME               TIME                   MIN   MAX    DESIRED
```

```
wii-scale-up       2010-04-05T02:00:00Z   200   200
wii-sale-over      2010-04-06T00:00:00Z   100   100
```

Updating Scheduled Actions

Use `PutScheduledUpdateGroupAction` to update an Auto Scaling group's scheduled action. Make a request with this Auto Scaling API action with a preexisting scheduled action plan name to overwrite the existing plan with any new values you specify for minimum, maximum, and desired group size.

Suspending Scheduled Actions

Use the `SuspendProcesses` action to suspend an Auto Scaling group's scheduled action. This action doesn't delete your scheduled actions. Make a request with this Auto Scaling API action with the name of the scheduled action plan you want deleted.

```
% as-suspend-processes my-group
```

Deleting Scheduled Actions

Use the `DeleteScheduledAction` action to delete an Auto Scaling group's scheduled action. Make a request with this Auto Scaling API action with the name of the scheduled action you want to delete.

Note

Scheduled actions are automatically deleted when they are executed.

Managing Scheduled Actions

You can determine whether instances were launched due to a scheduled action by examining the `Cause` field returned by `DescribeAutoScalingActivities`. Activities launched as a direct result of a scheduled action will have a reference to the specific action name in the cause field of the corresponding scaling activity. An example cause would be:

```
Cause => “At 2010-03-24T01:23:17Z scheduled action scale-up-come-morning set
group desired capacity changing the minimum capacity from 100 to 200.  At 2010-
03-24T01:24:12Z an instance was started in response to a difference between
desired and actual capacity, increasing the capacity from 100 to 200.”
```

Programming Considerations

- Auto Scaling guarantees execution order 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 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 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 will be rejected with an error message noting the conflict.

Auto Scaling and AWS Identity and Access Management

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, or Auto Scaling groups and triggers.

For general information about IAM, go to:

- Identity and Access Management (IAM)
- AWS Identity and Access Management Getting Started Guide
- Using AWS Identity and Access Management

Using IAM with Auto Scaling

Topics

- Auto Scaling ARNs (p. 29)
- Auto Scaling Actions (p. 29)
- Auto Scaling Keys (p. 29)
- Example Policies for Auto Scaling (p. 29)

Auto Scaling does not offer its own resource-based permissions system. However, Auto Scaling integrates with IAM so that you can specify which Auto Scaling 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 (e.g., a specific AutoScalingGroup). For example, you could create a policy that gives the Managers group permission to use only `DescribeAutoScalingGroups`, `DescribeLaunchConfigurations`, `DescribeScalingActivites`, and `DescribeTriggers`. They could then use those actions with any AutoScalingGroups and LaunchConfigurations 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 policies that cover Auto Scaling actions and resources, see Example Policies for Auto Scaling (p. 29).

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 a policy to control access to Auto Scaling actions, you use * as the resource. For more information about ARNs, 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` (for date/time conditions)
- `aws:EpochTime` (the date in epoch or UNIX time, for use with date/time conditions)
- `aws:SecureTransport` (Boolean representing whether the request was sent using SSL)
- `aws:SourceIp` (the requester's IP address, for use with IP address conditions)
- `aws:UserAgent` (information about the requester's client application, for use with string conditions)

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Example Policies for Auto Scaling

This section shows several simple 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 policies, based on the policy’s stated goals.

Example 1: Allow a group to create and manage AutoScaling LaunchConfigurations

In this example, we create a 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 LaunchConfigurations. 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.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":"autoscaling:*LaunchConfiguration*",
      "Resource":"*"
      }
   ]
}
```

Example 2: Allow system administrators to create and manage AutoScalingGroups and triggers

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

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["autoscaling:*Scaling*","autoscaling:*Trigger*"],
      "Resource":"*"
      }
   ]
}
```

Example 3: Allow developers to change the capacity of an AutoScalingGroup

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

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":"autoscaling:SetDesiredCapacity",
      "Resource":"*"
      }
   ]
}
```

Using Auto Scaling with AWS Products

Topics

- Amazon CloudWatch Monitoring for Auto Scaling Instances (p. 31)
- Auto Scaling Group Metrics (p. 33)
- Using CloudWatch Alarms to Execute Scaling Policies (p. 34)
- Using Amazon Virtual Private Cloud (p. 34)
- Using Spot Instances (p. 35)

You can use Auto Scaling with several other AWS products to create applications.

Amazon CloudWatch Monitoring for Auto Scaling Instances

Topics

- Activating Detailed Instance Monitoring for Auto Scaling (p. 31)
- Activating Basic Instance Monitoring for Auto Scaling (p. 32)

This section discusses the metrics that Auto Scaling instances send to Amazon CloudWatch. Instance metrics are the metrics that an individual Amazon EC2 instance sends to Amazon CloudWatch. Instance metrics are the same metrics available for any Amazon EC2 instance, whether or not it is in an Auto Scaling group.

Amazon CloudWatch offers basic or detailed monitoring. Basic monitoring sends aggregated data about each instance to Amazon 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. 33).

The following sections describe how to enable either detailed monitoring or basic monitoring.

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 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 Amazon 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 command line tools, create a launch configuration with the *`--monitoring-enabled`* option.
2. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will enable detailed monitoring for new instances that it creates.
3. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call `MonitorInstances` from the Amazon EC2 API for each existing instance to enable detailed monitoring.
Terminate existing instances	Call `TerminateInstanceInAutoScalingGroup` from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring enabled.

4. If you have Amazon CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the Amazon 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`. If you are using the command line tools, create a launch configuration with the *`--monitoring-disabled`* option.

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 command line tools, 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. 33).
3. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will disable detailed monitoring for new instances that it creates.
4. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call `UnmonitorInstances` from the Amazon EC2 API for each existing instance to disable detailed monitoring.
Terminate existing instances	Call `TerminateInstanceInAutoScalingGroup` from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring disabled.

5. If you have Amazon CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the Amazon 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 will 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 on instance metrics for Amazon EC2 instances, go to Amazon CloudWatch Developer Guide.

Auto Scaling Group Metrics

Group metrics are metrics that Auto Scaling group sends to Amazon CloudWatch to describe the group rather than any of its instances. If you enable group metrics, Auto Scaling sends aggregated data to Amazon CloudWatch every minute. If you disable group metrics, Auto Scaling does not send any group metrics data to Amazon 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 Amazon CloudWatch Monitoring for Auto Scaling Instances (p. 31).
2. Call `EnableMetricsCollection`, which is part of the Auto Scaling Query API. Alternatively, you can use the equivalent `as-enable-metrics-collection` command that is part of the Auto Scaling command line tools.

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. Units: *Count*
`GroupMaxSize`	The maximum size of the Auto Scaling group. Units: *Count*
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain. Units: *Count*
`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. Units: *Count*
`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. Units: *Count*
`GroupTerminatingInstances`	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending. Units: *Count*

Metric	Description
`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. Units: *Count*

Dimensions for Auto Scaling Group Metrics

The only dimension that Auto Scaling sends to Amazon CloudWatch is the name of the Auto Scaling group. This means that all available statistics are filtered by Auto Scaling group name.

Using CloudWatch Alarms to Execute Scaling Policies

CloudWatch alarms are a way to monitor a CloudWatch metric and to take automated action when the metric breaches a certain threshold. For more information about CloudWatch alarms, go to the CloudWatch Developer Guide.

When an alarm is fired, it sends a notification to Auto Scaling. Auto Scaling receives that message and executes the appropriate action for the Auto Scaling group. This behavior is stored in the Auto Scaling group's respective policy. A policy lets you specify the magnitude and type of change in the number of instances in an Auto Scaling group when its alarm fires.

For more information on how to use CloudWatch Alarms with Auto Scaling, see Auto Scaling With Alarms And Load Balancing (p. 45).

Using Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (VPC) enables you to designate your own private resources in the AWS cloud, and then connect those resources directly to your own data center using a VPN connection. You can use Auto Scaling to create Auto Scaling groups in your Amazon VPC. The Auto Scaling group instances created this way are isolated from instances that belong to other EC2 users.

To use Auto Scaling in a VPC, create your VPC and choose a subnet to contain your Auto Scaling group. You will use this subnet’s identifier when you create the Auto Scaling group.

When your VPC and its related objects are in place, you can create individual EC2 instances and Auto Scaling groups. For information on how to create individual EC2 instances, see Using Amazon Virtual Private Cloud in the *Amazon EC2 Developer Guide*.

To create an Auto Scaling group, specify a launch configuration as you normally would, and then create an Auto Scaling group that specifies your subnet identifier. You can specify that the Auto Scaling group launches instances in multiple subnets of the Amazon VPC. Use the `VPCZoneIdentifier` parameter available in both the `CreateAutoScalingGroup` and `UpdateAutoScalingGroup` actions to specify your subnet identifier. For more information about launching Auto Scaling instances into Amazon VPC, go to ???.

Important

Do NOT use the VPC identifier to specify the `VPCZoneIdentifier` parameter. Virtual Private Clouds have 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.

For more information about Amazon VPC and creating a VPC and subnets, go to the Amazon Virtual Private Cloud Getting Started Guide.

Using Spot Instances

If you have flexiblity about when your application runs, or your application can horizontally scale up with additional temporary computing resources, take a look at Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances. Using Spot Instances, you can bid on unused Amazon EC2 compute capacity (instances). You can run those instances for as long as your bid exceeds the current Spot price (market price). Amazon EC2 sets the Spot price and adjusts it periodically as requests come in and the available supply of instances changes. You place a Spot Instance request by specifying the maximum price (your bid price) you are willing to pay per hour per instance. If the maximum price of your bid is greater than the current Spot price, your request is fulfilled, and your instances run until you terminate them or the Spot price increases above your maximum price (whichever comes first). Everyone pays the same Spot price for that period regardless of whether their maximum bid price was higher, and you will never pay more than your hourly maximum bid price. For more information, go to Spot Instances in the *Amazon Elastic Compute Cloud User Guide*.

Auto Scaling integrates with Spot Instances to provide you with the ability to automate the capacity management of the groups of your Amazon EC2 Spot Instances. When you use Auto Scaling groups to launch your Spot Instances, you can take advantage of Auto Scaling features, such as dynamic scaling based on CloudWatch metrics, Amazon SNS notifications when your Spot Instances are added to or deleted from an Auto Scaling group, health checks of your spot instances, and other scaling policies such as schedule-based scaling.

To use Auto Scaling for your Spot Instances, you first create a *launch configuration* by specifying a Spot price (your bid price) for the Spot Instances to launch. Next, you create an Auto Scaling group using the launch configuration that has your Spot price. Auto Scaling will then use your Spot price to bid for your instances. If the price set by Amazon EC2 (market price) is lower than your Spot price, your bid will be fulfilled and your Spot Instances launched. You pay the market price for your instances as long as they are running or until the Spot price is higher than your maximum price.

For detailed instructions on using Auto Scaling to launch your Spot Instances, go to Using Auto Scaling to Launch Spot Instances (p. 50).

Using Auto Scaling

Topics

- Basic Scenario in Auto Scaling (p. 36)
- Auto Scaling With Email Notifications (p. 40)
- Auto Scaling with Alarms and Load Balancing (p. 45)
- Using Auto Scaling to Launch Spot Instances (p. 50)
- Launching Auto Scaling Instances with an IAM Role (p. 61)
- Expand to an Additional Availability Zone (p. 65)
- Merge into a Single Multi-Zone Group (p. 66)
- Shut Down an Auto-Scaled, Load-Balanced Application (p. 69)
- Suspend and Resume Processes (p. 73)
- Tagging Auto Scaling Groups and Amazon EC2 Instances (p. 74)

This section discusses some common scenarios for using Auto Scaling. In each scenario, we describe the situation, identify the pieces you need and why, list the tools you need and how to use them, and show you the steps to reach the goal. For a complete description of Auto Scaling actions, go to the API Reference.

Note

We assume that your instances are in the US Standard Region. 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-west-1* Region by using the `--region eu-west-1` parameter or setting the `EC2_REGION` environment variable.

Basic Scenario in Auto Scaling

You can use Auto Scaling in a number of different ways for example, you can maintain the number of instances you have running, or add or remove instances based on demand patterns. For any application in which you plan to use Auto Scaling, you must use certain building blocks to get started. In this section, we give you a basic scenario for setting up the infrastructure that will get Auto Scaling started for most applications.

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.

Tools You Will Use

You will use the following command line tool commands.

Command	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the specified Auto Scaling group(s) if the group exists.

For common conventions the documentation uses to represent command symbols, see Document Conventions.

Create a Launch Configuration

The launch configuration specifies the template that Auto Scaling uses to launch Amazon EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. In the following example, you will use the `as-create-launch-config` CLI command. You can also use the `CreateLaunchConfiguration` API call. For more information about the API call, see CreateLaunchConfiguration. For information about launch configuration, see Launch Configuration.

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

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

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

- Launch configuration name: `MyLC`
- Instance type: `m1.small`
- Image ID: `ami-0535d66c`

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.

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

```
as-create-launch-config MyLC --image-id ami-0535d66c --instance-type m1.small
```

Auto Scaling returns the following:

```
OK-Created launch config
```

You now have a launch configuration called MyLC that launches an `m1.small` instance using the `ami-0535d66c` 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 Amazon 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.

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, go to CreateAutoScalingGroup in the *Auto Scaling API Reference*. For information about Auto Scaling groups, see Auto Scaling Group.

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.

For this launch configuration, make the following specifications:

- Auto Scaling group name: `MyGroup`
- Launch configuration name: `MyLC`
- 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, go to the Amazon EC2 product page.

Enter the `as-create-auto-scaling-group` command.

```
as-create-auto-scaling-group MyGroup --launch-configuration MyLC --availability-
zones us-east-1a --min-size 1 --max-size 10 --desired-capacity 1
```

Auto Scaling returns the following:

```
OK-Created AutoScalingGroup
```

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

Verify Auto Scaling Group Creation

You use the `as-describe-auto-scaling-groups` command to confirm that the `MyGroup` 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-describe-auto-scaling-groups MyGroup --headers
```

Auto Scaling returns the following:

```
AUTO-SCALING-GROUP  GROUP-NAME   LAUNCH-CONFIG  AVAILABILITY-ZONES  MIN-SIZE
MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  MyGroup      MyLC           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 `MyGroup` 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-describe-auto-scaling-instances --headers
```

Auto Scaling returns the following (your `INSTANCE-ID` will differ):

```
INSTANCE  INSTANCE-ID  GROUP-NAME   AVAILABILITY-ZONE  STATE      STATUS
LAUNCH-CONFIG
INSTANCE  i-bcdd63d1   MyGroup      us-east-1a         InService  HEALTHY  MyLC
```

Note

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

You can also perform these tasks using the API calls `DescribeAutoScalingGroup` and `DescribeAutoScalingInstances`, respectively. In addition, you can use `DescribeScalingActivities` to get information about the scaling activities for your Auto Scaling group. For more information, go to the Auto Scaling API Reference.

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. So, if you run the snippet as a single code block, you will not see see the instance information right away.

```
as-create-launch-config MyLC --image-id ami-0535d66c --instance-type m1.small
as-create-auto-scaling-group MyGroup --launch-configuration MyLC --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
```

Auto Scaling With Email Notifications

You want to know right away when your Auto Scaling group changes that is, when Auto Scaling launches or terminates instances in your group.

Starting with API version 2011-01-01, you can use an Amazon Simple Notification Service (Amazon SNS)-backed Auto Scaling feature that sends notifications each time specified events take place. To enable this feature, you will need the following:

- An Amazon Resource Name (ARN), which you generate when you create an Amazon Simple Notification Service (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. In this section, we assume that you already have created an Amazon SNS topic and that you have an ARN. For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.

- An Auto Scaling Group, which you created when you went through Create an Auto Scaling Group (p. 38).
- 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. We will discuss the steps in setting up a notification configuration in the topic Set Up the Notification Configuration (p. 42) later in this section. 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 will cause the notification to be sent. The following table lists the available notification types:

Notification type	Event
`autoscaling:EC2_INSTANCE_LAUNCH`	Successful instance launch by Auto Scaling.
`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`	Failed instance launch by Auto Scaling.
`autoscaling:EC2_INSTANCE_TERMINATE`	Successful instance termination by Auto Scaling.
`autoscaling:EC2_INSTANCE_TERMINATE_ERROR`	Failed instance termination by Auto Scaling.
`autoscaling:TEST_NOTIFICATION`	Validate a configured SNS topic (as a result of calling PutNotificationConfiguration action)

For the most updated list of notification types, use the `as-describe-auto-scaling-notification-types` CLI command or the `DescribeAutoScalingNotificationTypes` API. For more information, go to the DescribeAutoScalingNotificationTypes in the *Auto Scaling API Reference*.

For information on troubleshooting `EC2_INSTANCE_LAUNCH_ERROR`, see Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure.

In this section, we set up SNS to send email notifications. When Auto Scaling launches instances to reach or maintain desired capacity, as specified in your Auto Scaling group, SNS sends a notification to your email address with information about the instances. You can check your email Inbox for this information, then run `as-describe-auto-scaling-group` to get information about current instances in the Auto Scaling group and confirm that the instances listed in your email actually exist.

Tools You Will Use

We walk you through the steps of the basic scenario using the command line tools. You can also use the API actions that correspond to the command line calls. For more information, go to the Auto Scaling API Reference.

You will be using the following command line tool commands.

Command	Description
`as-put-notification-configuration`	Configures an Auto Scaling group to send notifications when specified events take place.
`as-describe-notification-configurations`	Returns a list of notification actions associated with Auto Scaling groups for specified events.

Command	Description
`as-describe-auto-scaling-notification-types`	Returns a list of all notification types that are supported by Auto Scaling.
`as-delete-notification-configuration`	Deletes notifications created by `PutNotificationConfiguration`.
`as-delete-auto-scaling-group`	Deletes the specified Auto Scaling group.
`as-set-desired-capacity`	Sets the desired capacity of the Auto Scaling group.

Set Up the Notification Configuration

After you've created your Amazon SNS topic and you have the ARN, you are ready to set up the notification configuration. To configure your Auto Scaling group to send notifications when specified events take place, use `as-put-notification-configuration`.

The `as-put-notification-configuration` command takes the following arguments:

`as-put-notification-configuration` ***`AutoScalingGroupName`*** **`--notification-types`** ***`value`*** **`--topic-arn`** ***`topic-ARN`*** **`[General Options]`**

You need to specify the Auto Scaling group name, the ARN, and the notifcation types.

For this example, specify:

- Auto Scaling group name: `MyGroup`
- ARN:`arn:placeholder:MyTopic`

  Note

  ARNs are unique identifiers for Amazon Web Services (AWS) resources. Replace the ARN placeholder with your ARN.
- Notification types: `autoscaling:EC2_Instance_Launch, autoscaling:EC2_Instance_Terminate`

Open a command prompt and enter the `as-put-notification-configuration` command.

```
as-put-notification-configuration MyGroup --topic-arn arn:placeholder:MyTopic
--notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN
STANCE_TERMINATE
```

Auto Scaling returns the following:

```
OK-Put Notification Configuration
```

You now have a notification configuration that sends a notification to the endpoint subscribed in the `arn:placeholder:MyTopic` ARN whenever instances are launched or terminated.

Verify Notification Configuration

To verify the notification actions associated with your Auto Scaling group when specified events take place, use `as-describe-notification-configurations`.

The `as-describe-notification-configurations` command takes the following arguments:

`as-describe-notification-configurations [--auto-scaling-groups` *`value`*`[`*`,value...`*`]] [--maxrecords` *`value`*`] [General Options]`

If you specify the Auto Scaling group, this command returns a full list of all notification configuration 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. For this example, specify:

- Auto Scaling group name: `MyGroup`

Open a command prompt and enter the `as-describe-notification-configurations` command.

```
as-describe-notification-configurations --auto-scaling-groups MyGroup -headers
```

Auto Scaling returns the following:

```
NOTIFICATION-CONFIG  GROUP-NAME  TOPIC-ARN  NOTIFICATION-TYPE-NAME
NOTIFICATION-CONFIG  MyGroup  arn:placeholder:MyTopic  autoscaling:EC2_IN
STANCE_LAUNCH
NOTIFICATION-CONFIG  MyGroup  arn:placeholder:MyTopic  autoscaling:EC2_IN
STANCE_TERMINATE
```

You have confirmed that you have a notification configuration set up for the `MyGroup` Auto Scaling group.

Update and Generate Notifications

To cause the changes that will generate notifications, let's update the Auto Scaling group by changing the desired capacity of the `MyGroup` Auto Scaling group from 1 instance to 5 instances. When Auto Scaling launches the EC2 instances to the new desired capacity, SNS will send an email notification.

For this example, you will use `as-set-desired-capacity` command to change the desired capacity of the Auto Scaling group, and the `as-describe-auto-scaling-groups` to verify the instances of your Auto Scaling group.

The `as-set-desired-capacity` command takes the following arguments:

`as-set-desired-capacity` *`AutoScalingGroupName`* `--desired-capacity` *`value`* `[--honor-cooldown|no-honor-cooldown] [General Options]`

To use `as-set-desired-capacity`, you must specify the Auto Scaling group name (`MyGroup`) and the new number of instances as the desired capacity (5).

Open a command prompt and enter the commands.

```
as-set-desired-capacity MyGroup --desired-capacity 5
```

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 instances for `MyGroup` were launched. When you receive this notification, you can confirm it by using `as-describe-auto-scaling-groups` and specifying `MyGroup` as the Auto Scaling group name.

Delete Notification Configuration

Here's how you clean up when you're done: Delete the notification configuration, which we discuss in this section, and then delete the Auto Scaling group, which we discuss in the next section.

To delete the notification configuration, use `as-delete-notification-configuration`. The `as-delete-notification-configuration` command takes the following arguments:

`as-delete-notification-configuration` ***`AutoScalingGroupNames`*** **`--topic-ARN`** ***`value`*** **`[General Options]`**

Both the Auto Scaling group name and ARN are required. For this example, specify:

- Auto Scaling group name: `MyGroup`
- ARN:`arn:placeholder:MyTopic`

Open a command prompt and enter the command:

```
as-delete-notification-configuration MyGroup --topic-ARN arn:placeholder:MyTopic
```

After confirming that you want to delete the notification configuration, Auto Scaling returns the following:

```
OK-Deleted Notification Configuration
```

Delete Auto Scaling Group

To delete the Auto Scaling group, use `as-delete-auto-scaling-group`. Starting with API version 2011-01-01, you can use the new `--force-delete` argument to delete the Auto Scaling group and terminate all instances associated with it with one call. Previously, you would perform these two tasks by setting the desired capacity to zero (0), waiting until all instances are deleted, and then deleting the Auto Scaling group.

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

`as-delete-auto-scaling-group` ***`AutoScalingGroupNames`*** **`[--force-delete] [General Options]`**

In this example, we will use the optional `--force-delete` argument, so specify:

- Auto Scaling group name: `MyGroup`
- `--force-delete`

Open a command prompt and enter the commands:

```
as-delete-auto-scaling-group MyGroup --force-delete
```

After confirming that you want to delete the Auto Scaling group, Auto Scaling returns the following:

```
OK-Deleted Auto Scaling group
```

You have just deleted your notification configuration, instances, and Auto Scaling group. All that's left is the launch configuration you created in the Create a Launch Configuration section. To delete launch configurations, use the `as-delete-launch-config` CLI command with the launch configuration name.

The command looks like this:

```
as-delete-launch-config MyLC
```

For information on how to delete the SNS topic, go to DeleteTopic in the *Amazon Simple Notification Service Getting Started Guide*.

Tasks Completed

You just performed the following tasks:

- Set up a notification configuration
- Updated and generated notifications
- Deleted a notification configuration
- Deleted an Auto Scaling group

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.

```
as-put-notification-configuration MyGroup --topic-arn arn:placeholder:MyTopic
--notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN
STANCE_TERMINATE
as-describe-notification-configurations MyGroup -headers
as-set-desired-capacity MyGroup --desired-capacity 5
as-describe-auto-scaling-groups MyGroup
as-delete-notification-configuration MyGroup --topic-ARN arn:placeholder:MyTopic

as-delete-auto-scaling-group MyGroup --force-delete
```

Auto Scaling with Alarms and Load Balancing

In this example, you set up an Amazon EC2 application to be load-balanced and auto-scaled with a minimum number of two instances and maximum number of twenty instances. Auto Scaling in this example is configured to scale out by one when the application’s average CPU usage exceeds a threshold of 80 percent for 10 minutes, and scale in by one when the application's average CPU usage drops below 40 percent for 10 minutes. You use Amazon CloudWatch alarms to alert the application when a threshold is breached.

This example assumes that you have the following:

- An Amazon Machine Image (AMI) for your Amazon EC2 application

- A defined LoadBalancer named `MyLB` configured for Availability Zones `us-east-1a` and `us-east-1b`
- Installed Amazon CloudWatch command line tools if you're planning on using the command line interface.

Note

For information on how to create an AMI, go to Creating Your Own AMIs in the *Amazon Elastic Compute Cloud User Guide*. For more information about Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide. For information on installing the Amazon CloudWatch command line tool, see Command Line Tools.

Note

This scenario is for a load-balanced, auto-scaled application. In the following examples, if you aren't using Elastic Load Balancing, omit the list of LoadBalancers from step 2.

API Example

To set up an auto-scaled, load-balanced Amazon EC2 application

1. Call `CreateLaunchConfiguration` with the following parameters:
   - *`ImageId`* = *`ami-xxxxxxxx`*
   - *`LaunchConfigurationName`* = `MyLC`
   - *`InstanceType`* = `m1.small`

2. Call `CreateAutoScalingGroup` with the following parameters:
   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`
   - *`AvailabilityZones`* = `us-east-1a, us-east-1b`
   - *`LaunchConfigurationName`* = `MyLC`
   - *`LoadBalancerNames`* = `MyLB`
   - *`MaxSize`* = `20`
   - *`MinSize`* = `2`

3. Call `PutScalingPolicy` with the following parameters:
   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`
   - *`PolicyName`* = `MyScaleUpPolicy`
   - *`ScalingAdjustment`* = `1`
   - *`AdjustmentType`* = `ChangeInCapacity`
   - *`Cooldown`* = `300`

   Note

   The `PutScalingPolicy` action will return an ARN that represents your policy. Use your policy ARN, contained in the `PolicyARN` response element, to associate your policy with the CloudWatch alarm that you will create in the following step.

4. Call `PutMetricAlarm` from the CloudWatch API with the following parameters:

- *AlarmName* = `MyHighCPUAlarm`
- *AlarmActions* = *POLICY-ARN_from_previous_step*
- *MetricName* = `CPUUtilization`
- *Namespace* = `AWS/EC2`
- *Statistic* = `Average`
- *Period* = `600`
- *EvaluationPeriods* = `1`
- *Threshold* = `80`
- *ComparisonOperator* = `GreaterThanThreshold`
- *Dimensions* = `(Name=AutoScalingGroupName, Value=MyAutoScalingGroup)`

5. Call `PutScalingPolicy` again with the following parameters:
   - *AutoScalingGroupName* = `MyAutoScalingGroup`
   - *PolicyName* = `MyScaleDownPolicy`
   - *ScalingAdjustment* = `-1`
   - *AdjustmentType* = `ChangeInCapacity`
   - *Cooldown* = `300`

   Note

   The `PutScalingPolicy` action will return an ARN that represents your scale-down policy. Use your policy ARN, contained in the `PolicyARN` response element, to associate your policy with the CloudWatch alarm that you will create in the following step.
6. Call `PutMetricAlarm` from the CloudWatch API with the following parameters:
   - *AlarmName* = `MyLowCPUAlarm`
   - *AlarmActions* = *POLICY-ARN_from_previous_step*
   - *MetricName* = `CPUUtilization`
   - *Namespace* = `AWS/EC2`
   - *Statistic* = `Average`
   - *Period* = `600`
   - *EvaluationPeriods* = `1`
   - *Threshold* = `40`
   - *ComparisonOperator* = `LessThanThreshold`
   - *Dimensions* = `(Name=AutoScalingGroupName, Value=MyAutoScalingGroup)`

Your Amazon EC2 application has been launched as an auto-scaled and load-balanced application.

Command Line Tools Example

In this example, you will need to enter commands for both Auto Scaling and Amazon CloudWatch.

To set up an auto-scaled, load-balanced Amazon EC2 application

1. Use the Auto Scaling `as-create-launch-config` command.

   ```
   PROMPT>as-create-launch-config MyLC --image-id ami-xxxxxxxx --instance-type
    m1.small
   ```

   Auto Scaling returns the following:

   ```
   OK-Created launch config
   ```

2. Use the Auto Scaling `as-create-auto-scaling-group` command.

   ```
   PROMPT>as-create-auto-scaling-group MyAutoScalingGroup --launch-configuration
    MyLC --availability-zones us-east-1a us-east-1b  --min-size 2 --max-size
   20 --load-balancers MyLB
   ```

   Auto Scaling returns the following:

   ```
   OK-Created AutoScalingGroup
   ```

3. Use the Auto Scaling `as-put-scaling-policy` command to create a policy to enlarge your fleet.

   ```
   PROMPT>as-put-scaling-policy MyScaleUpPolicy --auto-scaling-group MyAutoScal
   ingGroup  --adjustment=1 --type ChangeInCapacity  --cooldown 300
   ```

   Auto Scaling returns the following (example output):

   ```
   POLICY-ARN arn:aws:autoscaling:us-east-1:0123456789:scalingPolicy/abc-1234-
   def-567
   ```

   Note

   The `as-put-scaling-policy` command returns an ARN that represents your policy. Use your policy ARN to associate your policy with the CloudWatch alarm that you will create in the following step.

4. Use the CloudWatch `mon-put-metric-alarm` command, substituting your `POLICY-ARN` for the `--alarm-actions` parameter.

   ```
   PROMPT>mon-put-metric-alarm MyHighCPUAlarm  --comparison-operator  Greater
   ThanThreshold  --evaluation-periods  1 --metric-name  CPUUtilization  --
   namespace  "AWS/EC2"  --period  600  --statistic Average --threshold  80 -
   -alarm-actions POLICY-ARN_from_previous_step --dimensions "AutoScalingGroup
   Name=MyAutoScalingGroup"
   ```

   Amazon CloudWatch returns the following:

   ```
   OK-Created Alarm
   ```

5. Use the Auto Scaling `as-put-scaling-policy` command to create a policy to reduce your fleet size. If you are using Windows, wrap the `--adjustment` parameter in quotation marks: `"--adjustment=-1"`.

   ```
   PROMPT>as-put-scaling-policy MyScaleDownPolicy --auto-scaling-group MyAuto
   ScalingGroup  --adjustment=-1 --type ChangeInCapacity  --cooldown 300
   ```

   Auto Scaling returns the following (example output):

   ```
   POLICY-ARN arn:aws:autoscaling:us-east-1:0123456789:scalingPolicy/abc-1234-
   def-567
   ```

   Note

   The `as-put-scaling-policy` command returns an ARN that represents your policy. Use your policy ARN to associate your policy with the CloudWatch alarm that you will create in the following step.

6. Use the CloudWatch `mon-put-metric-alarm` command, substituting the `MyScaleDownPolicy` `POLICY-ARN` for the ARN shown in the `--alarm-actions` parameter.

   ```
   PROMPT>mon-put-metric-alarm MyLowCPUAlarm  --comparison-operator  LessTh
   anThreshold --evaluation-periods  1 --metric-name  CPUUtilization --namespace
     "AWS/EC2"  --period  600  --statistic Average --threshold  40  --alarm-
   actions POLICY-ARN_from_previous_step --dimensions "AutoScalingGroup
   Name=MyAutoScalingGroup"
   ```

   Amazon CloudWatch returns the following:

   ```
   OK-Created Alarm
   ```

7. Verify that your Auto Scaling group exists by using the `as-describe-auto-scaling-groups` command.

   ```
   PROMPT>as-describe-auto-scaling-groups MyAutoScalingGroup --headers
   ```

   Auto Scaling returns the following:

   ```

   AUTO-SCALING-GROUP  GROUP-NAME          LAUNCH-CONFIG  AVAILABILITY-ZONES
       LOAD-BALANCERS  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
   AUTO-SCALING-GROUP  MyAutoScalingGroup  MyLC           us-east-1b,us-east-
   1a  MyLB            2         20        2
   INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
   INSTANCE  i-xxxxxxxx   us-east-1b         InService  Healthy  MyLC
   INSTANCE  i-xxxxxxxx   us-east-1a         InService  Healthy  MyLC
   ```

Your Amazon EC2 application has been launched as an auto-scaled and load-balanced application.

Using Auto Scaling to Launch Spot Instances

If you have flexibility about when to run your applications, or if the performance of your application will scale efficiently with the addition of temporary compute resources, and you want to benefit from the cost-savings of using Amazon EC2 Spot Instances, you can set up Auto Scaling to launch Spot Instances.

Spot Instances are a way for you to 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. For more information about Spot Instances, go to Spot Instances in the *Amazon Elastic Compute Cloud User Guide*.

When you use Spot Instances with Auto Scaling, first you need to understand the basics of Spot Instance bids 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 will terminate 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.
- **Maintaining your Spot Instances.** When your instance is terminated (whether it's a Spot Instance or an On-Demand instance), Auto Scaling will launch another instance in an effort to replace it and maintain the desired capacity specified in the Auto Scaling group. However, whether or not 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 will be launched; if the market price is higher than the bid price, then no instance will be 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, we will create a launch configuration and an Auto Scaling group that launches Spot Instances. We will use the Auto Scaling command line interface (CLI) to create the launch configuration and the Auto Scaling group, and to verify and obtain information about the new instances. In this scenario, we will perform the following tasks:

1. Create a launch configuration, including the `--spot-price` option to specify the bid price for the Spot Instances to launch.
2. Create an Auto Scaling group, specifying the launch configuration in which you specified a spot bid price.
3. Verify and check your Auto Scaling group and instances.
4. Set up notifications.
5. Update the bid price.
6. Schedule spot bids.
7. Clean up.

Tools You Will Use

You will use the following Auto Scaling command line interface (CLI) commands.

Command	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the Auto Scaling groups, if the groups exist.
`as-describe-auto-scaling-instances`	Describes the Auto Scaling instances, if the instances exist.
`as-describe-scaling-activities`	Describes a set of activities or all activities belonging to a group.
`as-put-notification-configuration`	Configures an Auto Scaling group to send notifications when specified events take place.
`as-describe-notification-configurations`	Returns a list of notification actions associated with Auto Scaling groups for specified events.
`as-put-scheduled-update-group-action`	Creates or updates a scheduled update group action.
`as-delete-auto-scaling-group`	Deletes the specified Auto Scaling group, if the group has no instances and no scaling activities are in progress.

For common conventions the documentation uses to represent command symbols, see Document Conventions.

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 (p. 36). 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 this 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 $0.03, or any other price below $0.05, and 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. So, if the current Spot market price drops to $0.02 or rises to $0.04, you will pay the new 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 Elastic Compute Cloud User Guide*.

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. 57) later in this walkthrough.

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

- Launch configuration name = `spotlc-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, see Amazon Machine Images (AMIs).
- 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 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. In addition, let Auto Scaling know at what level it should maintain your fleet of Spot Instances by setting `min-size` and `desired-capacity`. If your Spot bid price falls below the Spot price, and Amazon EC2 terminates your instance, Auto Scaling will submit your bid again and launch an instance in an effort to maintain the desired capacity you specified. However, whether or not 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 will be launched; if the market price is higher than the bid price, then no instance will be launched at that point. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 38).

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 like 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 like the following example:

```
OK-Created AutoScalingGroup
```

Verifying and Checking 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 will return 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 walkthrough, we are using the `as-describe-scaling-activities` command to see the state of your bid. Your command should look like 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 like the following example:

  ```
  ACTIVITY  ACTIVITY-ID                                   END-TIME              GROUP-
  NAME      CODE                          MESSAGE
  ACTIVITY  31bcbb67-7f50-4b88-ae7e-e564a8c80a90                                 spotasg
          WaitingForSpotInstanceId  Placed Spot instance request: sir-fc8a3014.
   Waiting for instance(s)
  ACTIVITY  770bbeb5-407c-404c-a826-856f65db1c57                                 spotasg
          WaitingForSpotInstanceId  Placed Spot instance request: sir-69101014.
   Waiting for instance(s)
  ACTIVITY  597e4ebd-220e-42bc-8ac9-2bae4d20b8d7  2012-05-23T17:40:22Z  spotasg
           Successful
  ```

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 is lower than the Spot market price.

If you run the `as-describe-scaling-activities` command again later, you can be getting information that is similar to the following example:

```
ACTIVITY  ACTIVITY-ID                           END-TIME              GROUP-
NAME      CODE
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 we 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 like 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 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 will launch 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, go to Maintaining Current Scaling Level (p. 21).

  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 will place the bid specified in the current launch configuration and attempt 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 will show details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see Create an Auto Scaling Group (p. 38).

  This is the basic syntax:

  as-describe-auto-scaling-groups [*AutoScalingGroupNames* [*AutoScalingGroupNames*...]] [--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 like 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-c630d1bf   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 will return 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 walkthrough, we are assuming 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 like 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-c630d1bf   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 may want to 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 Simple Notification Service (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 Simple Notification Service (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 Getting Started Guide*.
- An Auto Scaling Group, which you created earlier in this walkthrough.
- 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 will cause the notification to be sent.

For information about setting up email notifications in Auto Scaling, go to Auto Scaling With Email Notifications (p. 40).

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.

If, for example, you want to launch a set of Spot Instances that have a higher likelihood of running uninterrupted 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 in this walkthrough.

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 = spotlc-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 spotlc-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 walkthrough, the only change to the *spotasg* Auto Scaling group is the launch configuration 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 like the following example:

```
ACTIVITY  ACTIVITY-ID                           END-TIME             GROUP-
NAME      CODE                        MESSAGE
ACTIVITY  5879cc50-1e40-4539-a754-1cb084f1aecd                      spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-93828812.
 Waiting for instance(s)
ACTIVITY  777fbe1b-7a24-4aaf-b7a9-d368d0511878                      spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-016cf812.
 Waiting for instance(s)
ACTIVITY  f4b00f81-eaea-4421-80b4-a2e3a35cc782                      spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-cf60ea12.
 Waiting for instance(s)
ACTIVITY  31bcbb67-7f50-4b88-ae7e-e564a8c80a90                        spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-fc8a3014.
 Waiting for instance(s)
ACTIVITY  770bbeb5-407c-404c-a826-856f65db1c57                        spotasg
        WaitingForSpotInstanceId  Placed Spot instance request: sir-69101014.
 Waiting for instance(s)
ACTIVITY  597e4ebd-220e-42bc-8ac9-2bae4d20b8d7  2012-05-23T17:40:22Z  spotasg
         Successful

ACTIVITY  eca158b4-a6f9-4ec5-a813-78d42c1738e2  2012-05-23T17:40:22Z  spotasg
         Successful

ACTIVITY  1a5bd6c6-0b0a-4917-8cf0-eee1044a179f  2012-05-23T17:22:19Z  spotasg
         Successful

ACTIVITY  c285bf16-d2c4-4ae8-acad-7450655facb5  2012-05-23T17:22:19Z  spotasg
         Successful

ACTIVITY  127e3608-5911-4111-906e-31fb16d43f2e  2012-05-23T15:38:06Z  spotasg
         Successful

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 will continue running and will only be terminated if the Spot market price goes above the bid price on which the running Spot Instance was based.

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 Options]`**

In this walkthrough, 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 like the following example:

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

To check your scheduled action, run `as-describe-scheduled-actions`.

You will 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 will be 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 like 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. So, 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
```

Launching Auto Scaling Instances with an IAM Role

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

You can use IAM roles with Auto Scaling to automatically enable applications running on your Amazon 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 you create an IAM role that has all the permissions required to access the AWS resources, then you add your role to the instance profile.

For more information about IAM roles and instance profiles, go to Working With Roles in *Using IAM*.

Prerequisites: Using IAM

In this section, we walk you through the 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 information and instructions on creating and managing an IAM role using the IAM console, the IAM CLI, or the IAM Query API, go to Create a Role in *Using IAM*.

If you plan to use the IAM CLI, be sure to install the IAM CLI tool. For information about setting up and using the IAM CLI, go to AWS Identity and Access Management Command Line Interface Reference.

Steps for Launching Auto Scaling 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 you just created.
- Verify that the EC2 instance was launched with the IAM role.

Commands You Will Use

You will use the following Auto Scaling commands.

Commands	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with the specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the Auto Scaling groups, if the groups exist.

For common conventions the documentation uses to represent command symbols, see Document Conventions.

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 (p. 36). Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

For this walkthrough, 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, see Amazon Machine Images (AMIs).
- 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 detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 38).

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 like 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 like the following example:

```
OK-Created AutoScalingGroup
```

Verify that EC2 Instance Launches with 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 Create an Auto Scaling Group (p. 38).

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 will 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                  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 will be 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
```

Expand to an Additional Availability Zone

In this example, you expand a load-balanced application to an additional Availability Zone.

For this example, we assume that the application is auto-scaled and running in two Availability Zones (`us-east-1a` and `us-east-1b`).

API Example

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

1. Call `UpdateAutoScalingGroup` with the following parameters:
   - *`AvailabilityZones`* = `us-east-1a, us-east-1b, us-east-1c`
   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`
   - *`MinSize`* = `3`

   Note

   This scenario is for both a load-balanced and auto-scaled application. If you aren't using Elastic Load Balancing or don't want to load-balance your application, omit steps 2 and 3.
2. Call the Elastic Load Balancing service API `DescribeInstanceHealth`.
   - *`LoadBalancerName`* = `MyLB`

   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 Elastic Load Balancing and ready, proceed to the next step. For more information about Elastic Load Balancing, go to the Elastic Load Balancing Developer Guide.

   Note

   When you call `EnableAvailabilityZonesForLoadBalancer`, the LoadBalancer begins to route traffic equally among all the enabled Availability Zones.
3. Call `EnableAvailabilityZonesForLoadBalancer`:
   - *`AvailabilityZones`* = `us-east-1c`
   - *`LoadBalancerName`* = `MyLB`

You have now load-balanced your Amazon EC2 application across three Availability Zones and auto-scaled it in each zone.

Command Line Tools Example

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

1. Use the Auto Scaling `as-update-auto-scaling-group` command as in the following example.

```
PROMPT>as-update-auto-scaling-group MyAutoScalingGroup --availability-zones
 us-east-1a, us-east-1b, us-east-1c --min-size 3
```

Auto Scaling returns the following:

```
OK-Updated AutoScalingGroup
```

Note

This scenario is for both a load-balanced and auto-scaled application. If you aren't using Elastic Load Balancing or don't want to load balance your application, omit steps 2 and 3.

2. Use the Elastic Load Balancing `elb-describe-instance-health` command as in the following example.

```
PROMPT>elb-describe-instance-health  MyLB
```

Elastic Load Balancing returns the following:

```
INSTANCE  INSTANCE-ID STATE
INSTANCE  i-4f8cf126 InService
INSTANCE  i-0bb7ca62 InService
```

When the status of each of the new instances (in the Auto Scaling group in the new Availability Zone) changes to `InService`, indicating that the instances are now ready to accept traffic from Elastic Load Balancing, proceed to the next step.

When you call the `elb-enable-zones-for-lb` command, the LoadBalancer begins to route traffic equally among all of the enabled Availability Zones.

3. Use the Elastic Load Balancing `elb-enable-zones-for-lb` command, as in the following example.

```
PROMPT>elb-enable-zones-for-lb  MyLB  --headers --availability-zones us-
east-1c
```

Elastic Load Balancing returns the following:

```
AVAILABILITY_ZONES  AVAILABILITY-ZONES
AVAILABILITY_ZONES  "us-east-1a, us-east-1b, us-east-1c"
```

Merge 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.

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`. Each group is defined with two minimum instances, five maximum instances, and a desired capacity of three.

Note

This example works for groups with or without a LoadBalancer, provided that the new multi-zone group will be in one of the same zones as the original single-zone groups.

API Example

To merge separate single-zone Auto Scaling groups into a single multi-zone group

1. Call `UpdateAutoScalingGroup` with the following parameters to add `myGroupA` to additional Availability Zones:
   - *`AvailabilityZones`* = `us-east-1a, us-east-1c`
   - *`AutoScalingGroupName`* = `myGroupA`
   - *`MinSize`* = `4`
   - *`MaxSize`* = `10`
2. Call `SetDesiredCapacity` with the following parameters to increase the capacity of `myGroupA` to six:
   - *`AutoScalingGroupName`* = `myGroupA`
   - *`DesiredCapacity`* = `6`
3. Call `DescribeAutoScalingGroups` with the following parameter to check that `myGroupA` has been successfully added to the additional zones and is at the required capacity:
   - *`AutoScalingGroupName`* = `myGroupA`
4. Delete `myGroupB`:

   a. Call `UpdateAutoScalingGroup` with 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 `DescribeAutoScalingGroup` `myGroupB` to verify that no instances are left in the group.

      ii. Call `DescribeScalingActivites` `myGroupB` to verify that all scaling activities have completed.

   c. Call `DeleteAutoScalingGroup` with the following parameter:
   - *`AutoScalingGroupName`* = `myGroupB`

Command Line Tools Example

To merge separate single-zone Auto Scaling groups into a single multi-zone group:

1. Call the Auto Scaling `as-update-auto-scaling-group` command to add `myGroupA` to additional Availability Zones.

   ```
   PROMPT>as-update-auto-scaling-group myGroupA --availability-zones us-east-
   1a, us-east-1c –-max-size 10 –-min-size 4
   ```

   Auto Scaling returns the following:

   ```
   OK-AutoScaling Group updated
   ```

2. Call the `as-set-desired-capacity` command to increase the capacity of `myGroupA` to six.

   ```
   PROMPT>as-set-desired-capacity myGroupA --desired-capacity 6
   ```

3. Call 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:

   ```
   PROMPT>as-describe-auto-scaling-groups myGroupA
   ```

4. Delete `myGroupB`:

   a. Use the Auto Scaling `as-update-auto-scaling-group` command.

      ```
      PROMPT>as-update-auto-scaling-group myGroupB --min-size 0 --max-size 0
      ```

      Auto Scaling returns the following:

      ```
      OK-AutoScaling Group updated
      ```

   b. Verify that no instances remain in your Auto Scaling group by using the Auto Scaling `as-describe-auto-scaling-groups` command.

      Note

      Call this command until no more instances remain in the Auto Scaling group.

      ```
      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. Verify that your scaling activities are successful by using the Auto Scaling `as-describe-scaling-activities` command.

```
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 Auto Scaling `as-delete-auto-scaling-group` command.

```
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 trigger.

Auto Scaling returns the following:

```
OK-Deleted AutoScalingGroup
```

Shut Down an Auto-Scaled, Load-Balanced Application

In this example, you completely shut down an application that uses Auto Scaling, Elastic Load Balancing and Amazon CloudWatch alarms. For an example of how to set up an Auto Scaling application that uses Elastic Load Balancing and Amazon CloudWatch alarms, see Auto Scaling With Alarms And Load Balancing (p. 45).

All Amazon EC2 instances for an online application are shut down. All Auto Scaling groups and Amazon CloudWatch alarms associated with the application are deleted.

In this example, we assume you have an application (associated with a LoadBalancer named `MyLB`) running on a set of Amazon EC2 instances in an Auto Scaling group named `MyAutoScalingGroup`. The application is load-balanced, and is also auto-scaled through corresponding scaling triggers:

- `MyHighCPUAlarm` that initiates `MyScaleUpPolicy`
- `MyLowCPUAlarm` that initiates `MyScaleDownPolicy`

Note

This scenario is for an application that is both load-balanced and auto-scaled. If you aren't using Elastic Load Balancing, omit step 3 from the following examples.

API Example

To shut down an auto-scaled, load-balanced application

1. Delete `MyScalingGroup`:

    a. Call `UpdateAutoScalingGroup` with the following parameters:

    - *`AutoScalingGroupName`* = `MyAutoScalingGroup`
    - *`MinSize`* = `0`
    - *`MaxSize`* = `0`

    b. Verify that your changes to `MyAutoScalingGroup` have taken effect by doing the following:

    i. Call `DescribeAutoScalingGroups` with `MyAutoScalingGroup` as a request parameter to verify that there are no instances left in the group.

    ii. Call `DescribeScalingActivites` with `MyAutoScalingGroup` as a request parameter to verify that all scaling activities have completed.

    c. Call `DeleteAutoScalingGroup` with the following parameters:

    - *`AutoScalingGroupName`* = `MyAutoScalingGroup`

    Note

    Auto Scaling deletes any policies associated with the Auto Scaling group you are deleting.

    d. Call `DeleteLaunchConfiguration` with the following parameters:

    - *`LaunchConfigurationName`* = `MyLC`

2. Call `DeleteAlarms` from the Amazon CloudWatch API with the following parameters:

    - *`AlarmNames.member.1`* = `MyHighCPUAlarm`
    - *`AlarmNames.member.2`* = `MyLowCPUAlarm`

3. Delete the LoadBalancer.

    - Call `DeleteLoadBalancer` from the Elastic Load Balancing API with the following parameters:

    ```
    LoadBalancerName = MyLB
    ```

The application is completely shut down.

For documentation on *DeleteLoadBalancer*, refer to the API Reference of the Elastic Load Balancing Developer Guide.

Command Line Tools Example

To shut down an auto-scaled, load-balanced application

1. Delete `MyAutoScalingGroup`:

   a. Use the Auto Scaling `as-update-auto-scaling-group` command to change the minimum and maximum size.

```
PROMPT>as-update-auto-scaling-group MyAutoScalingGroup --min-size 0 --
max-size 0
```

   Auto Scaling returns the following:

```
OK-AutoScaling Group updated
```

   b. Verify that no instances are in your Auto Scaling group by using the Auto Scaling `as-describe-auto-scaling-groups` command.

```
PROMPT>as-describe-auto-scaling-groups MyAutoScalingGroup --headers
```

   Auto Scaling might report that the state of your instances is `Terminating` because the termination process can take a few minutes:

```
AUTO-SCALING-GROUP  GROUP-NAME           LAUNCH-CONFIG         AVAIL
ABILITY-ZONES  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  MyAutoScalingGroup   MyLC                 us-east-
1a          0         0         0
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE        STATUS   LAUNCH-
CONFIG
INSTANCE  i-xxxxxxxx   us-east-1b         Terminating  Healthy  MyLC
INSTANCE  i-xxxxxxxx   us-east-1a         Terminating  Healthy  MyLC
```

   After the termination process completes, calls to this command will report that no more instances remain in the Auto Scaling group:

```

AUTO-SCALING-GROUP  GROUP-NAME           LAUNCH-CONFIG         AVAIL
ABILITY-ZONES  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  MyAutoScalingGroup   MyLC                 us-east-
1a          0         0         0
```

   c. Use the Auto Scaling `as-delete-auto-scaling-group` command.

```
PROMPT>as-delete-auto-scaling-group MyAutoScalingGroup
```

   Auto Scaling returns the following:

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

Enter y to delete the AutoScalingGroup.

Auto Scaling returns the following:

```
OK-Deleted AutoScalingGroup
```

Note

Auto Scaling deletes any policies associated with the Auto Scaling group you are deleting.

d. Use the Auto Scaling `as-delete-launch-config` command to delete the launch configuration.

```
PROMPT>as-delete-launch-config MyLC
```

Auto Scaling returns the following:

```
Are you sure you want to delete this launch configuration? [Ny]
```

Enter y to delete the Launch Configuration.

Auto Scaling returns the following:

```
OK-Deleted launch configuration
```

2. Delete the Amazon CloudWatch alarms.

   - Use the Amazon CloudWatch `mon-delete-alarms` command.

```
PROMPT>mon-delete-alarms MyHighCPUAlarm MyLowCPUAlarm
```

Auto Scaling returns the following:

```
Are you sure you want to delete these alarms?  [Ny]y
```

Enter y to delete the alarms.

Auto Scaling returns the following:

```
OK-Deleted alarms
```

3. Delete the load balancer with the Elastic Load Balancing `elb-delete-lb` command.

```
PROMPT>elb-delete-lb  MyLB
```

Elastic Load Balancing returns the following:

```
Warning: Deleting a LoadBalancer can lead to service disruption to any cus
tomers
connected to the LoadBalancer. Are you sure you want to delete
this LoadBalancer? [Ny]
```

Enter y to delete the LoadBalancer

Elastic Load Balancing returns the following:

```
OK-Deleting LoadBalancer
```

The application is completely shut down.

Suspend and Resume Processes

In this example, you suspend all scaling activities on an Auto Scaling group to investigate a configuration problem with your Amazon EC2 application. After the investigation concludes, you resume scaling activities on the Auto Scaling group.

For this example, assume that you have an Amazon EC2 application running within a single Auto Scaling group named `MyAutoScalingGroup`.

You suspect a configuration problem with your Amazon EC2 application, so you want to suspend scaling processes with the `SuspendProcesses` action while you investigate the problem. This strategy lets you make configuration changes that might otherwise trigger Auto Scaling activities.

Upon concluding your investigation, resume scaling processes with a call to the Auto Scaling action `ResumeProcesses`.

API Example

To suspend and then resume scaling processes on an auto-scaled, load-balanced Amazon EC2 application

1. Call `SuspendProcesses` with the following parameters:
   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`

2. Call `ResumeProcesses` with the following parameters:
   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`

Your Amazon EC2 application resumes normal Auto Scaling activities.

Command Line Tools Example

In this example, you will need to enter commands for both Auto Scaling and Elastic Load Balancing.

To suspend and resume processes on an Auto Scaling group

1. Use the Auto Scaling `as-suspend-processes` command.

   ```
   PROMPT>as-suspend-processes MyAutoScalingGroup
   ```

   Auto Scaling returns the following:

   ```
   OK-Processes Suspended
   ```

2. After concluding your investigation, use the Auto Scaling `as-resume-processes` action.

   ```
   PROMPT>as-resume-processes MyAutoScalingGroup
   ```

   Auto Scaling returns the following:

   ```
   OK-Processes Resumed
   ```

Your Amazon EC2 application has resumed normal Auto Scaling activities.

Tagging Auto Scaling Groups and Amazon EC2 Instances

You can use Auto Scaling group tags to organize your Auto Scaling resources and provide additional information for your Auto Scaling group such as software version, role, or location information. Like Amazon Elastic Compute Cloud (Amazon EC2) tags, Auto Scaling group tags provide search, group, and filter functionality. These tags have a key and value that can be modified. You can also remove Auto Scaling group tags any time. For more information about Amazon EC2 tags, see Using Tags.

Additional Features

Optionally, you can propagate Auto Scaling group tags to the Amazon EC2 instances launched by Auto Scaling. When you use the `PropagateAtLaunch` flag with the `as-create-or-update-tags` command, the tag you create will be applied to new Amazon EC2 instances launched by the Auto Scaling group. Likewise, when you modify a tag, the updated version will be applied to new instances launched by the Auto Scaling group after the change. Tags created or modified with the `as-create-or-update-tags` command will not apply to instances that are already running at the time you used the command. For an example, go to Change the Tag (p. 79).

> **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, cannot be edited or deleted, and do not count against your limit of 10 tags per instance.

In this section, we will use the Auto Scaling command line interface (CLI) commands to create and update tags. We will use Amazon EC2 CLI commands to confirm that the new and updated tags are applied to instances launched after the tags are created. In this scenario, we will perform the following tasks:

1. Create a launch configuration.
2. Create an Auto Scaling group with a tag.
3. Confirm that the Auto Scaling group has a tag and that the tag is applied to instances that the Auto Scaling group launches.
4. Change the tag and the Auto Scaling group, and confirm that the changes are propagated to the new instances that the Auto Scaling group launches.
5. Clean up.

Tools You Will Use

You will use the following Auto Scaling and Amazon EC2 command line interface (CLI) commands.

Note

We will discuss in detail the Auto Scaling CLI commands. For more information on specific Amazon EC2 CLI commands that we use in this walkthrough, go to the Amazon Elastic Compute Cloud CLI Reference.

Command	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with specified name and other attributes.
`as-create-or-update-tags`	Creates new tags or updates existing tags.
`as-describe-auto-scaling-groups`	Describes the Auto Scaling groups, if the groups exist.
`as-describe-auto-scaling-instances`	Describes the Auto Scaling instances, if the instances exist.
`as-describe-tags`	Lists your tags. You can filter the list to return only the tags you specify.
`as-update-auto-scaling-group`	Updates specified Auto Scaling group.
`ec2-describe-instances`	Returns information about EC2 instances that you own.
`as-delete-auto-scaling-group`	Deletes the specified auto scaling group, if the group has no instances and no scaling activities are in progress.

For common conventions the documentation uses to represent command symbols, see Document Conventions.

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 (p. 36). Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

For this tagging walkthrough, the following values will be specified for the `as-create-launch-config` command:

- Launch configuration name = `MyTagLC`
- Image ID = `ami-31814f58`
  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- Instance type = `m1.small`

Your command should look similar to the following example:

```
as-create-launch-config MyTagLC --image-id ami-31814f58 --instance-type m1.small
```

You should get a confirmation like the following example:

```
OK-Created launch config
```

Create a Tag for Your Auto Scaling Group

Create your Auto Scaling group with a tag and specify that the tag also applies to the instances launched by the Auto Scaling group. You use the `--tag` option to define the tag for your Auto Scaling group and instances. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 38).

Specify these values for the command:

- Auto Scaling Group name = `MyTagASG`
- Launch configuration name = `MyTagLC`
- Availability Zone = `us-east-1a`
- Max size = `10`
- Min size = `1`

In addition, the `--tag` parameter should be specified this way:

`--tag` "k=*`value`*, [id=*`value`*], [t=*`value`*], [v=*`value`*], [p=*`value`*]" [`--tag` "k=*`value`*, [id=*`value`*], [t=*`value`*], [v=*`value`*], [p=*`value`*]"...]

Provide these values for the following tag options:

Option	Description	Example value
k	Tag's key, as part of key-value pair. Always required.	`version`
v	Tag's value, as part of key-value pair. Optional.	`1.0`

Option	Description	Example value
`t`	The type of resource to which the tag is applied. Not required with `as-create-auto-scaling-group` because the tag is being created with the Auto Scaling group resource.	Not specified with the `as-create-auto-scaling-group` command.
`id`	The name of the resource to which the tag is applied. Not required with `as-create-auto-scaling-group` because the tag is being created with the Auto Scaling group resource.	Not specified with the `as-create-auto-scaling-group` command.
`p`	The propagate-at-launch flag. Specify this flag only if you want the tags to be applied to newly launched Amazon EC2.	`true`

Your command should look like the following example:

```
as-create-auto-scaling-group MyTagASG --launch-configuration MyTagLC --availab
ility-zones us-east-1a --min-size 1 --max-size 10 --tag "k=version, v=1.0,
p=true"
```

You should get confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Confirm Tag Creation

To verify that the tag is created, applied to the Auto Scaling group, and that the instance is launched, use the following Auto Scaling commands:

- `as-describe-tags`
- `as-describe-auto-scaling-groups`
- `as-describe-auto-scaling-instances`

In addition, use the `ec2-describe-instances` EC2 command to get detailed information about the instances that the Auto Scaling group launched.

1. 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 walkthrough, specify the following:

   - Tag key: `key=version`
   - Tag value: `value=1.0`

Your command should look like the following example:

```
as-describe-tags --filter "key=version, value=1.0"
```

You should receive a confirmation that Auto Scaling created the tag, like the following example response:

```
TAG MyTagASG auto-scaling-group version 1.0 true
```

2. Use the `as-describe-auto-scaling-groups` command to confirm that the tag is applied to the Auto Scaling group `MyTagASG`. The command describes the Auto Scaling group, if the group exists, and takes the following arguments:

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

   Your command should look like the following example:

```
as-describe-auto-scaling-groups MyTagASG
```

   You should receive a response that includes information confirming that Auto Scaling applied the tag to the Auto Scaling group `MyTagASG`. The response should look like the following example:

```
AUTO-SCALING-GROUP  MyTagASG  MyTagLC  us-east-1a   1  10   5
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
TAG  RESOURCE-ID  RESOURCE-TYPE       KEY      VALUE  PROPAGATE-AT-LAUNCH
TAG  MyTagASG     auto-scaling-group  version  1.0    true
```

3. Use the `as-describe-auto-scaling-instances` command to get a list of EC2 instance that the Auto Scaling group launched. This command describes the specified Auto Scaling instances, if they exist, or lists all the Auto Scaling instances if no instance is specified. The command takes the following arguments:

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

   For this walkthrough, do not specify any of the optional parameters, so your command should look like the following example:

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

   You should get a listing that looks like the following example:

```
INSTANCE  i-33698556   MyTagASG   us-east-1b    InService  Healthy  MyTagLC
```

   From the result, get the instance ID of the instance that the `MyTagASG` launched. You will use it for the next command.

4. Use the `ec2-describe-instances` command to get specific information about EC2 instances. This command returns information about instances you own. For more information about this CLI command, see ec2-describe-instances.

   In the `ec2-describe-instances` command, specify the instance ID of the EC2 instance to which the tag was propagated. Your command should look like the following example:

```
ec2-describe-instances i-33698556
```

Your result should show that the tag was applied to the instance, as in the following example:

```
RESERVATION r-45253524 629715795501 default
INSTANCE i-33698556 ami-31814f58 ec2-23-20-37-36.compute-1.amazonaws.com ip-
10-32-153-166.ec2.internal running  0  m1.small 2012-01-23T21:41:44+0000 us-
east-1a aki-805ea7e9   monitoring-enabled 23.20.37.36 10.32.153.166   ebs
  paravirtual xen 29849241-1242-4524-ba07-09248cd29652 sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-2f303942 2012-01-23T21:42:12.000Z
TAG instance i-33698556 version 1.0
TAG instance i-33698556 aws:autoscaling:groupName MyTagASG
```

Change the Tag

Now you can update the tag and modify the Auto Scaling group. Use the `as-create-or-update-tags` command and the `as-update-auto-scaling-group` command to make the changes.

The `as-create-or-update-tags` 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]`

Update the tag value to `2.0`, so your command should look like the following example:

```
as-create-or-update-tags --tag "id=MyTagASG, t=auto-scaling-group, k=version,
v=2.0, p=true"
```

Note

You must specify the resource name and resource type so that Auto Scaling knows to which Auto Scaling resource the tag applies. Resource name is the *`id`* (in the example it's `id=MyTagASG`) and resource type is *`t`* (in the example, it's `t=auto-scaling-group`).

You should receive a confirmation similar to the following example:

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

Now, you can use the `as-update-auto-scaling-group` command to modify the Auto Scaling group by increasing the number of EC2 instances to be launched. Change the `min-size` value to 5.

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

`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]`**

Your command should look similar to the following example:

```
as-update-auto-scaling-group MyTagASG --min-size 5
```

The confirmation should look like the following example:

```
OK-Updated AutoScalingGroup
```

To verify that Auto Scaling updated the Auto Scaling Group `MyTagASG` and that Auto Scaling launched four additional instances to meet the new minimum size of 5, run the `as-describe-auto-scaling-instances` command with no arguments, like the following example:

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

Your results should be similar to the following example:

```
INSTANCE  i-2346af46  MyTagASG          us-east-1a  InService  HEALTHY  MyTagLC
INSTANCE  i-2546af40  MyTagASG          us-east-1a  InService  HEALTHY  MyTagLC
INSTANCE  i-33698556  MyTagASG          us-east-1a  InService  HEALTHY  MyTagLC
INSTANCE  i-5346af36  MyTagASG          us-east-1a  InService  HEALTHY  MyTagLC
INSTANCE  i-566da836  MyAutoScalingGroup  us-east-1b  InService  HEALTHY  MyL
Cwebapp
INSTANCE  i-5946af3c  MyTagASG          us-east-1a  InService  HEALTHY  MyTagLC
```

To verify that tag value 2.0 was applied to instances launched after the tag was updated, run the `ec2-describe-instances` command.
You should get a result set similar to the following example:

```
RESERVATION r-45253524 629715795501 default
INSTANCE i-33698556 ami-31814f58 ec2-23-20-37-36.compute-1.amazonaws.com ip-10-
32-153-166.ec2.internal running  0  m1.small 2012-01-23T21:41:44+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 23.20.37.36 10.32.153.166   ebs     para
virtual xen 29849241-1242-4524-ba07-09248cd29652 sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-2f303942 2012-01-23T21:42:12.000Z
TAG instance i-33698556 version 1.0
TAG instance i-33698556 aws:autoscaling:groupName MyTagASG
RESERVATION r-b1ebf9d0 629715795501 default
INSTANCE i-5346af36 ami-31814f58 ec2-107-22-159-5.compute-1.amazonaws.com ip-
10-204-54-176.ec2.internal running  0  m1.small 2012-01-24T07:17:50+0000 us-
east-1a aki-805ea7e9   monitoring-enabled 107.22.159.5 10.204.54.176   ebs
 paravirtual xen a848fb19-703f-4579-9ee4-930b2f1e5a9f sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-b3aea6de 2012-01-24T07:18:10.000Z
TAG instance i-5346af36 aws:autoscaling:groupName MyTagASG
TAG instance i-5346af36 version 2.0
RESERVATION r-b7ebf9d6 629715795501 default
INSTANCE i-5946af3c ami-31814f58 ec2-50-16-11-77.compute-1.amazonaws.com ip-10-
118-54-85.ec2.internal running  0  m1.small 2012-01-24T07:17:52+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 50.16.11.77 10.118.54.85   ebs     paravir
tual xen bfa8068e-a3a9-4374-8db0-7550010f477f sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-8daea6e0 2012-01-24T07:18:15.000Z
TAG instance i-5946af3c version 2.0
TAG instance i-5946af3c aws:autoscaling:groupName MyTagASG
RESERVATION r-bbebf9da 629715795501 default
INSTANCE i-2546af40 ami-31814f58 ec2-50-19-152-11.compute-1.amazonaws.com ip-
10-116-241-173.ec2.internal running  0  m1.small 2012-01-24T07:17:53+0000 us-
east-1a aki-805ea7e9   monitoring-enabled 50.19.152.11 10.116.241.173   ebs
  paravirtual xen 7d419c3a-0527-4970-80f5-b6467ce44849 sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-89aea6e4 2012-01-24T07:18:18.000Z
```

```
TAG instance i-2546af40 version 2.0
TAG instance i-2546af40 aws:autoscaling:groupName MyTagASG
RESERVATION r-bfebf9de 629715795501 default
INSTANCE i-2346af46 ami-31814f58 ec2-184-72-167-153.compute-1.amazonaws.com ip-
10-203-54-74.ec2.internal running  0  m1.small 2012-01-24T07:17:54+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 184.72.167.153 10.203.54.74   ebs
paravirtual xen ec0221df-0274-44e1-aa13-04b7f17442ef sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-8baea6e6 2012-01-24T07:18:15.000Z
TAG instance i-2346af46 aws:autoscaling:groupName MyTagASG
TAG instance i-2346af46 version 2.0
```

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 will be 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 manually stopped and terminated all instances of that Auto Scaling group.

Run the command with the following values:

- Auto Scaling group name = `MyTagASG`
- Force delete (optional parameter) = `--force-delete`

Your command should look like the following example:

```
as-delete-auto-scaling-group MyTagASG --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.
- Created an Auto Scaling group with a tag.
- Confirmed that the Auto Scaling group has a tag and that the tag is applied to instances that the Auto Scaling group launched.
- Changed the tag and Auto Scaling group, and confirmed that the changes are propagated to new instances launched.
- 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. So, 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 MyTagLC --image-id ami-31814f58 --instance-type m1.small
as-create-auto-scaling-group MyTagASG --launch-configuration MyTagLC --availab
ility-zones us-east-1a --min-size 1 --max-size 10 --tag "k=version, v=1.0,
p=true"
as-describe-tags --filter "key1=version, key2=1.0"
as-describe-auto-scaling-groups MyTagASG
as-describe-auto-scaling-instances
ec2-describe-instances i-33698556
as-create-or-update-tags --tag "id=MyTagASG, t=auto-scaling-group, k=version,
v=2.0, p=true"
as-update-auto-scaling-group MyTagASG --min-size 5
as-describe-auto-scaling-instances
ec2-describe-instances
as-delete-auto-scaling-group MyTagASG --force-delete
```

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, go to 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 will have to install the Amazon EC2 command line tools before you can use them. For more information go to Getting Started with the Command Line Tools in the *Amazon Elastic Compute Cloud User Guide*. For information on using the Query API, go to Making API Requests in the *Amazon Elastic Compute Cloud User Guide*.

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

Issues with	Error Message
Auto Scaling group	AutoScalingGroup <Auto Scaling group name> not found. (p. 87)
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request ...... (p. 87)
AWS account	You are not subscribed to this service. Please go to http://aws.amazon.com. (p. 87)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 87)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 88)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 88)
Capacity groups activity	???
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>).... (p. 87)
Key pair	The key pair <key pair associated with your Amazon EC2 instance> does not exist. Launching EC2 instance failed. (p. 86)

Issues with	Error Message
Launch configuration	The requested configuration is currently not supported. (p. 86)
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 88)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 86)

Troubleshooting Auto Scaling: Amazon EC2 AMIs

Issues with	Error Message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 89)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 89)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 90)
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. 90)
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. 89)

Troubleshooting Auto Scaling: Load Balancer Configuration

Issues with	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 90)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 91)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 91)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 91)

Troubleshooting Auto Scaling: Capacity Limits

Issues with	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 92)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>).... (p. 92)

Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure

The following topics provide information about your Amazon EC2 instances that fail to launch, potential causes, and the steps you can take to resolve the issues.

When your Amazon EC2 instances fail to launch, you might get one or moreof 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. 83).

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 Amazon 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-kepairs 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 go to http://aws.amazon.com.

- **Cause**: Your AWS account might have expired.
- **Solution**:Go to http://aws.amazon.com and click the **Sign Up Now** button 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 on the device name format, go to Instance Store Device Names in the *Amazon Elastic Compute Cloud User Guide*.
  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 on block device mapping, see Block Device Mapping.
  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 on valid instance types supported by the the cluster placement groups, go to Using Cluster Instances in the *Amazon Elastic Compute Cloud User Guide*.
  2. Follow 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 images, potential causes, and the steps you can take to resolve the issues.

When your Amazon 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. 83).

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

This 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 Amazon 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. 83).

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, go to 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 Amazon 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 the **Sign Up Now** button 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 Amazon 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. 83).

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, go to Contact Us and place a request to raise your Amazon EC2 instance limit.

Glossary

Access Key ID	An alphanumeric token that uniquely identifies a request sender. This ID is associated with your Secret Access Key.
administrative suspension	Auto Scaling might suspend processes for Auto Scaling groups that repeatedly fail to launch instances. Auto Scaling groups that most commonly experience administrative suspension have zero running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time in launching any instances.
Amazon Elastic Compute Cloud	The Amazon Elastic Compute Cloud (Amazon EC2) is a web service that enables you to launch and manage server instances in Amazon's data centers using APIs or available tools and utilities.
Amazon Machine Image	An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon Simple Storage Service (Amazon S3). It contains all the information necessary to boot instances of your software.
Amazon Simple Queue Service	Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers.
Availability Zone	Amazon EC2 locations are composed of Regions and Availability Zones. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity to other Availability Zones in the same Region.
Auto Scaling group	Auto Scaling key term. A representation of an application running on multiple Amazon Elastic Compute Cloud (EC2) instances. For more information, see Auto Scaling Group (p. 5).
breach	Auto Scaling term describing the condition in which a user-set threshold (upper or lower boundary) is passed. If the duration of the breach is determined to be significant, set by a breach duration parameter, then the trigger fires and possibly performs a scaling activity.
cooldown	Auto Scaling term. A period of time after a trigger is fired during which no other trigger activity can take place. A cooldown period allows the effect of a scaling activity to become visible in the metrics that originally triggered the activity. This period is configurable, and gives the system time to perform and adjust to any new scaling activities (such as scale-in and scale-out) that affect capacity.
dimension	Amazon CloudWatch key term. A *dimension* is part of the unigue identifier of a *metric*. It specifies how CloudWatch aggregates data. For more information, go to the *Amazon CloudWatch Developer Guide*.

EC2 instance	A virtual computer running in the cloud. EC2 instances are launched from an Amazon Machine Image (AMI).
launch configuration	Auto Scaling key term. The parameters used to create new EC2 instances.
life cycle	Auto Scaling term that refers to the life cycle state of the EC2 instances contained in an Auto Scaling group. EC2 instances progress through several states over their lifespan; these include *Pending*, *InService*, *Terminating* and *Terminated*.
LoadBalancer	Elastic Load Balancing key term. A LoadBalancer is represented by a DNS name and provides the single destination to which all requests intended for your application should be directed. For more information, go to the *Elastic Load Balancing Developer Guide*.
metric	Amazon CloudWatch key term. A metric is an aggregation of values from selected measures stored in one-minute time slots. For more information, go to the *Amazon CloudWatch Developer Guide*.
Region	Amazon EC2 locations are composed of Regions and Availability Zones. Regions are geographically dispersed and are in separate geographic areas or countries. Regions consist of one or more Availability Zones.
Scaling Activity	Auto Scaling key term. A process that changes the size of an Auto Scaling group. For more information, see Scaling Activity (p. 6).
Secret Access Key	Amazon Web Services (AWS) key used for request authentication.
statistic	Amazon CloudWatch key term. A *statistic* is a time-series of values for a metric. For more information, go to the *Amazon CloudWatch Developer Guide*.
trigger	Auto Scaling key term. The mechanism used to initiate Auto Scaling activities.
unbounded	Term used in Web Service Definition Language (WSDL), i.e., `maxOccurs="unbounded"`, meaning that the number of potential occurrences is not limited by a set number. Often used when defining a data type that is a list of other types, such as an unbounded list of integers (element members) or an unbounded list of other complex types that are element/members of the list being defined.
unit	Amazon CloudWatch key term. Every measure that is received by the CloudWatch Service has a unit attached, such as seconds or bytes. For more information,go to the *Amazon CloudWatch Developer Guide*.

Document History

The following table describes the important changes to the *Auto Scaling Developer Guide*.

- API version: 2011-01-01
- Latest documentation update: June 11, 2012

Change	Description	Release Date
Running Spot Instances	You can now run Spot Instances in Auto Scaling groups by specifying a Spot Instance bid price in your launch configuration. For a brief overview of Spot Instances, see Using Spot Instances (p. 35). For instructions on launching Spot Instances using Auto Scaling groups, see Using Auto Scaling to Launch Spot Instances (p. 50).	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. 83).	15 May 2012
Tagging Auto Scaling Groups and Amazon EC2 Instances	You can now tag Auto Scaling groups and specify that the tag also applies to Amazon EC2 instances that the Auto Scaling group launches after the tag is created. For more information, see Tagging Auto Scaling Groups and Amazon EC2 Instances (p. 74).	26 January 2012

Change	Description	Release Date
SNS Integration and Other Features	Auto Scaling now supports Amazon Simple Notification Services (Amazon SNS) so that you can use it to receive notifications whenever Auto Scaling launches or terminates Amazon EC2 instances. For more information, see Auto Scaling With Email Notifications (p. 40). Auto Scaling also added the following new features: • 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 up without adding the launched instance to the load balancer (LoadBalancer). For more information, see the ProcessType API datatype. • The `ForceDelete` flag in the `DeleteAutoScalingGroup` 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
New link	This service's endpoint information is now located in the Amazon Web Services General Reference. For more information, go to Regions and Endpoints in Amazon Web Services General Reference.	2 March 2011
Updated example	Added a missing *`Dimensions`* parameter to examples that call the `PutMetricAlarm` command. For more information, see Auto Scaling with Alarms and Load Balancing (p. 45).	12 January 2011
New feature	Added the ability to create scheduled scaling actions. For more information, see Using Auto Scaling (p. 36).	2 December 2010
New feature	Added support for Amazon Virtual Private Cloud (VPC). For more information, see Using Amazon Virtual Private Cloud (p. 34).	2 December 2010
New feature	Added support for high performance computing (HPC) clusters.	2 December 2010
New feature	Added the capability to use Elastic Load Balancing Health Check as the health check for Auto Scaling-managed Amazon EC2 instances. For more information, see Maintaining Current Scaling Level (p. 21).	2 December 2010
New design	Removed the older Trigger mechanism and redesigned Auto Scaling to use the CloudWatch alarm feature. For more information, see Using Auto Scaling (p. 36).	2 December 2010

Change	Description	Release Date
New feature	Added a new feature that lets you suspend and resume scaling processes. For more information, see Suspendable Processes (p. 7).	2 December 2010
New feature	This service now integrates with AWS Identity and Access Management (IAM). For more information, see Auto Scaling and AWS Identity and Access Management (p. 28).	2 December 2010
New Region	Auto Scaling now supports the Asia Pacific (Singapore) Region. For more information, see Auto Scaling Group (p. 5) and Endpoints (p. 17).	28 April 2010

Index

A

account ID
    getting, 15
API
    Query, 17
    User Scenarios, 36, 40, 45, 61, 65, 66, 69, 73, 74
ARNs
    for Auto Scaling, 29
authentication
    Query, 18
    signature version 2, 18
Auto Scaling, 28
    Amazon CloudWatch dimensions available, 34
    Auto Scaling metrics available, 33
    conceptual overview, 3
    Features, 2
    Spot Instances, 35
    Virtual Private Cloud (VPC), 34
Auto Scaling group, 5
Availability Zones and Regions
    conceptual overview, 8

C

certificates
    creating, 15
Conceptual Overviews
    Auto Scaling, 3
    Availability Zones and Regions, 8
    Suspendable Processes, 7
Cooldown, 7
credentials
    using, 15

D

data types
    RequestId, 18
Dimensions
    dimensions available for Auto Scaling group metrics, 34

E

error, 83
    retrieving, 83

G

glossary, 93

H

Health Check, 5
health check, 21
    configuring, 21
    customizing, 21
    deleting, 21
    listing, 21
    updating, 21

K

key terms
    alarm, 6
    Auto Scaling group, 5
    Health Check, 5
    Launch Configuration, 5
    Policy, 6
    Scaling Activity, 6
    Scheduled Update, 6
    Tagging, 5
    Trigger, 5

L

Launch Configuration, 5
login
    getting, 15

M

Metrics
    Auto Scaling group metrics, 33

O

Overviews
    What Is Auto Scaling?, 2

P

password
    using, 15
policies
    examples, 29
Policy, 6
policy
    creating, 23
    deleting, 23
    listing, 23
    suspending, 23

Q

Query
    API, 17
    authentication, 18
    parameters, 18

R

Regions, 13
RequestId, 18
ResponseMetadata
    RequestId, 18

S

scaling
    configuring, 21
scaling action, 21
Scaling Activity, 6
scheduled action
    creating, 26
    deleting, 26
    listing, 26
    suspending, 26
    updating, 26
Scheduled Update, 6
signature version 2, 18
suspend/restart, 83

T

Tagging Auto Scaling Groups, 5
Trigger, 5
troubleshooting, 83

U

User Scenarios
    Auto Scaling With Alarms and Load Balancing, 45
    Auto Scaling With Email Notifications, 40
    Expand to an Additional Availability Zone, 65
    Merge Into a Single Multi-Zone Group , 66
    Setting Up Auto Scaling Groups, 36
    Shut Down Auto-Scaled, Load-Balanced Application, 69
    Suspend and Resume Processes, 73
    Tagging Auto Scaling Groups and Instances, 74
    Using IAM Role For EC2 Instances, 61
using
    credentials, 15