Auto Scaling

Developer Guide

API Version 2010-08-01

Auto Scaling: Developer Guide

Table of Contents

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...?

How Do I...	Relevant Resources
Learn more about the business case for Auto Scaling	Auto Scaling product information
Learn how Auto Scaling works	What Is 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	User Scenarios (p. 37)
Configure Auto Scaling	Configuring Auto Scaling
Control and manage user identity and access	Auto Scaling and AWS Identity and Access Management (p. 10)
Get the technical FAQ	Auto Scaling Technical FAQ
Get help from the community of developers	Amazon EC2 Discussion Forums

What Is Auto Scaling?

Topics

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

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.

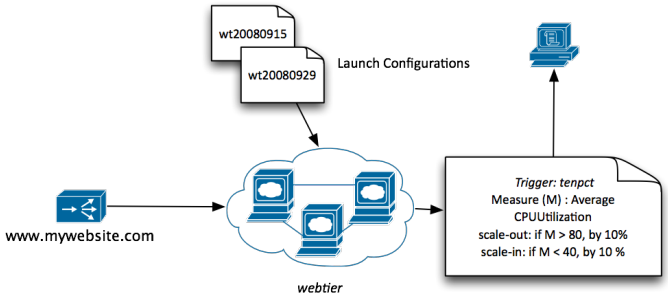
Auto Scaling lets you automatically adjust the size of a fleet of EC2 instances. Auto Scaling can add or remove EC2 instances from the fleet to help you seamlessly deal with load changes to your application, and let you introduce your own plans so you can proactively set the size of your Auto Scaling 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. 13), 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. 24).

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*.

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 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...
AlarmNotifications	Ignores all Amazon CloudWatch notifications.

If you suspend...	Auto Scaling...
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; at this time, Amazon has four Regions: the US-East (Northern Virginia) Region (also known as the US Standard Region), the US-West (Northern California) Region, the Asia Pacific (Singapore) Region, and the EU (Ireland) Region. 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.

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 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 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. 11).

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

Topics

This section provides task-oriented descriptions of how to use and implement Auto Scaling actions. For a complete description of Auto Scaling actions, go to the API Reference.

Using the Command Line Tools

Topics

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

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 `as-api-tools-A.B-nnnn` (`A`, `B`, and `n` 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. 19).
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 Eastern United States Region (`us-east-1`) with the `autoscaling.us-east-1.amazonaws.com` service endpoint URL. This section describes how to specify a different Region by changing the service endpoint URL.

To specify a different Region

1. View available Regions by entering the following:

```
PROMPT> ec2-describe-regions
REGION  ap-southeast-1       autoscaling.ap-southeast-1.amazonaws.com
```

```
REGION  eu-west-1            autoscaling.eu-west-1.amazonaws.com
REGION  us-east-1            autoscaling.us-east-1.amazonaws.com
REGION  us-west-1            autoscaling.us-west-1.amazonaws.com
```

2. If you want to change the service endpoint, set the `AWS_AUTO_SCALING_URL` environment variable as follows:
   - 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

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. 18).
- **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. 19).
- **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. 19).
- **Account ID**—Used to share resources with other AWS accounts. For more information, see Viewing Your Account ID (p. 20).

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 email 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. 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 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

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 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=2010-08-01
&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=2010-08-01
```

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=2010-08-01
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&AWSAccessKeyId=<Your AWS Access Key ID>
&Signature=<URLEncode(Base64Encode(Signature))>
&Expires=2011-02-10T12%3A00%3A00Z
```

Configuring Auto Scaling

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`:

```
% as-update-autoscaling-group myGroup –-health-check-type 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-autoscaling-group myGroup –-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 –-health-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`:

```
# When scaling up, double my group's current capacity
%as-put-scaling-policy my-group --name "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 --name "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 --name double-group-size
--adjustment 100 --type PercentOfCapacity --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 --name "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  GROUP-NAME  POLICY-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
desired 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
terminated 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

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.

Using Auto Scaling with AWS Products

Topics

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

Amazon CloudWatch Monitoring for Auto Scaling Instances

Topics

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.

To...	Do This...
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*

Metric	Description
`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*
`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. 37).

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. Use the `VPCZoneIdentifier` parameter—available in both the `CreateAutoScalingGroup` and `UpdateAutoScalingGroup` actions—to specify your subnet identifier.

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.

Troubleshooting Applications

Topics

Auto Scaling provides specific and descriptive errors to help you troubleshoot problems.

Retrieving Errors

Typically, you want your application to check if a request generated an error before processing results. The easiest way to find out if an error occurred is to look for an *`Error`* node in the response.

XPath syntax provides a simple way to search for the presence of an *`Error`* node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine whether an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with your Auto Scaling applications:

- Verify that your XSL style sheets are valid.

To do this, run your requests both with and without the XSL style sheet, to determine if the problem is in the request or in your style sheets.

- Check the structure of your request.
  Each Auto Scaling operation has a reference page. Double-check that you are using parameters correctly. Try the request on one of the other locales, if other locales exist. Try your request with SOAP through your browser. This will help you determine if the problem lies with your code or your request or with Auto Scaling.
- Look at the sample applications and use cases to see if those examples are performing similar operations.

User Scenarios

Topics

This section discusses some common user scenarios for Auto Scaling. These scenarios demonstrate the needed API and command line sequences to accomplish the given tasks.

Note

The examples in the following sections 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-west1* Region by using the `--region eu-west-1` parameter or setting the *`EC2_URL`* environment variable.

Auto Scaling with Alarms and Load Balancing

In this example, you set up an EC2 application to be load-balanced and auto-scaled with a minimum number of two instances and maximum number of 20 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 EC2 application
- A defined LoadBalancer named `MyLB` configured for Availability Zones `us-east-1a` and `us-east-1b`

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.

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 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 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 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
MyAutoScalingGroup  --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
GreaterThanThreshold  --evaluation-periods  1 --metric-name  CPUUtilization
  --namespace  "AWS/EC2"  --period  600  --statistic Average --threshold
80 --alarm-actions POLICY-ARN_from_previous_step --dimensions
"AutoScalingGroupName=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
MyAutoScalingGroup  --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
LessThanThreshold --evaluation-periods  1 --metric-name  CPUUtilization --
namespace  "AWS/EC2"  --period  600  --statistic Average --threshold  40
--alarm-actions POLICY-ARN_from_previous_step --dimensions
"AutoScalingGroupName=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 EC2 application has been launched as an auto-scaled and load-balanced application.

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 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 AutoScalingGroup 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 `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 load balancer, 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
      AVAILABILITY-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. 37).

All 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 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 load balancer.

   - 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
   AVAILABILITY-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
   AVAILABILITY-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
customers
connected to the LoadBalancer. Are you sure you want to delete
this LoadBalancer? [Ny]
```

Enter y to delete the Load Balancer

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 EC2 application. After the investigation concludes, you resume scaling activities on the Auto Scaling group.

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

You suspect a configuration problem with your 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 EC2 application

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

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

Your EC2 application has resumed 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.

How 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 EC2 application has resumed normal Auto Scaling activities.

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. This documentation is associated with the 2010-08-01 release of Auto Scaling. This guide was last updated on 14 July 2011.

Change	Description	Release Date
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.	In this release
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. 37).	12 January 2011
New feature	Added the ability to create scheduled scaling actions. For more information, see Using Auto Scaling (p. 13).	Current release
New feature	Added support for Amazon Virtual Private Cloud (VPC). For more information, see Using Amazon Virtual Private Cloud (p. 34).	Current release
New feature	Added support for high performance computing (HPC) clusters.	Current release
New feature	Added the capability to use Elastic Load Balancing Health Check as the health check for Auto Scaling-managed EC2 instances. For more information, see Maintaining Current Scaling Level (p. 24).	Current release
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. 13).	Current release
New feature	Added a new feature that lets you suspend and resume scaling processes. For more information, see Suspendable Processes (p. 7).	Current release

Change	Description	Release Date
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. 10).	Current release
New Region	Auto Scaling now supports the Asia Pacific (Singapore) Region. For more information, see Auto Scaling Group (p. 5) and Endpoints (p. 20).	28 April 2010

Document Conventions

This section lists the common typographical and symbol use conventions for AWS technical publications.

Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere. You can use this resource regularly. 
Code in text	Inline code samples (including XML) and commands are identified with a special font. You can use the command `java -version`.

Code blocks	Blocks of sample code are set apart from the body and marked accordingly.
	`# ls -l /var/www/html/index.html`
	`-rw-rw-r--  1 root root 1872 Jun 21 09:33 /var/www/html/index.html`
	`# date`
	`Wed Jun 21 09:33:42 EDT 2006`

Emphasis	Unusual or important words and phrases are marked with a special font. You *must* sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked.
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code. If the value is `null`, the returned response will be `false`.

Convention	Description/Example
Product and feature names	Named AWS products and features are identified on first use. Create an Amazon Machine Image (AMI).
Operations	In-text references to operations. Use the `GetHITResponse` operation.
Parameters	In-text references to parameters. The operation accepts the parameter *`AccountID`*.
Response elements	In-text references to responses. A container for one `CollectionParent` and one or more `CollectionItems`.
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the *Amazon Mechanical Turk Developer Guide*.
User entered values	A special font marks text that the user types. At the password prompt, type **`MyPassword`**.
User interface controls and labels	Denotes named items on the UI for easy identification. On the **File** menu, click **Properties**.
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. `% ec2-register <your-s3-bucket>/image.manifest` See also the following symbol convention.

Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses | and | vertical | bars)	Within a code description, bar separators denote options from which one must be chosen. `% data = hdfread (start | stride | edge)`

Optional parameters
XML variable text
	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters.
		`% sed [-n, -quiet]`
		Use square brackets in XML examples to differentiate them from tags.
		`<CustomerId>[ID]</CustomerId>`

Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value. `% ec2-register <your-s3-bucket>/image.manifest`

Index

T

U