Auto Scaling

Developer Guide

API Version 2011-01-01

Amazon Web Services

Auto Scaling: Developer Guide

Amazon Web Services

What is Auto Scaling?

Topics

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. Amazon EC2 instances are basically servers in the cloud. For applications configured to run on a cloud infrastructure, scaling is an important part of cost control and resource management. *Scaling* is the ability to increase or decrease the compute capacity of your application either by changing the number of servers (horizontal scaling) or changing the size of the servers (vertical scaling).

In a typical business situation, when your web application starts to get more traffic, you either add more servers or increase the size of your existing servers to handle the additional load. Similarly, if the traffic to your web application starts to slow down, you either terminate the under-utilized servers or decrease the size of your existing servers. Depending on your infrastucture, vertical scaling might involve changes to your server configurations every time you scale. With horizontal scaling, you simply increase or decrease the number of servers according to your application's demands. The decision when to scale vertically and when to scale horizontally depends on factors such as your use case, cost, performance, and infrastrucure.

When you scale using Auto Scaling you can increase the number of servers you're using automatically when the user demand goes up to ensure that performance is maintained, and you can decrease the number of servers when demand goes down to minimize costs. Auto Scaling helps you make efficient use of your compute resources by automatically doing the work of scaling for you. This automatic scaling is the core value of the Auto Scaling service.

Auto Scaling is well suited for applications that experience hourly, daily, or weekly variability in usage and need to automatically scale horizontally to keep up with usage variability. Auto Scaling frees you from having to accurately predict huge traffic spikes and plan for provisioning resources in advance of them. With Auto Scaling you can build a fully scalable and affordable infrastructure on the cloud.

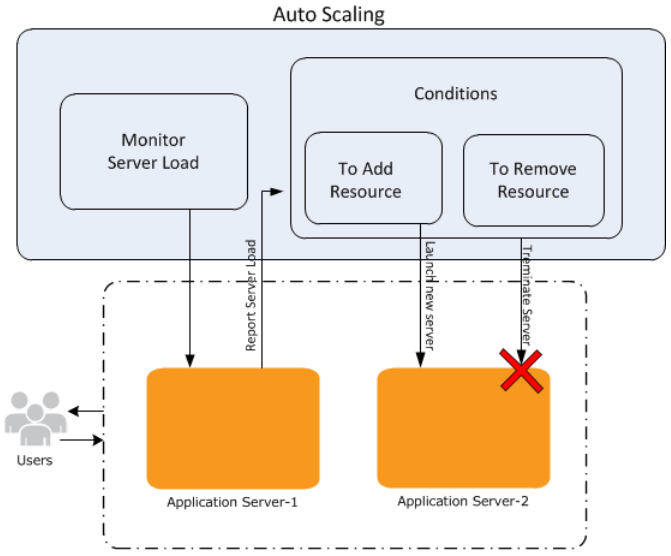
Auto Scaling allows you to scale your compute resources dynamically and predictably:

- Dynamically based on conditions specified by you (for example, increasing CPU utilization of your Amazon EC2 instance)
- Predictably according to a schedule defined by you (for example, every Friday at 13:00:00).

Let's look at an example of how Auto Scaling works. Suppose you have a web application that runs on a single cloud server. The single server performs well when you have regular traffic. However, occasionally the traffic to your application increases up to three times the normal load. When that happens, you need an additional cloud server to handle the increased traffic. For your application to scale gracefully with the additional traffic load, you'll need to launch the second cloud server ahead of the time when the increased load occurs, and then terminate that server after traffic goes down to normal levels. This process works best where your application has predictable traffic patterns, so that you'll know when to launch the additional server and when to terminate it.

But what if you do not know when the next traffic spike will hit your application? Where traffic spikes are not possible to predict, you would need to launch two cloud servers and keep them running at all times, even when the second server rarely gets any traffic. Of course, the additional server will incur costs while it is running.

What happens in this example if you use Auto Scaling? First, you will not have to keep the second server running all the time. Instead, you define the conditions that determine the increasing traffic to your application servers, and then tell Auto Scaling to launch a similar application server whenever those conditions are met. Second, you define another set of conditions that determine the decreasing traffic to your application servers and then tell Auto Scaling to terminate a server when those conditions are met. The following diagram illustrates a set of simple Auto Scaling conditions.



How Auto Scaling Works

In a common web application scenario, you run multiple copies of your application simultaneously to cover the volume of your customer traffic. These multiple copies of your application are hosted on identical Amazon EC2 instances (cloud servers), each handling customer requests. Auto Scaling manages the launch and termination of these EC2 instances on your behalf. This section assumes that you are familiar with Amazon Elastic Compute Cloud (Amazon EC2), and that you are using EC2 instances. For information about EC2 instances, see Amazon EC2 Instances.

When you use Auto Scaling, your EC2 instances are categorized into Auto Scaling *groups* for the purposes of instance scaling and management. You create Auto Scaling groups by defining the minimum, maximum, and, optionally, the desired number of running EC2 instances the group must have at any point in time.

Your Auto Scaling group uses a *launch configuration* to launch EC2 instances. You create the launch configuration by providing information about the image you want Auto Scaling to use to launch EC2

instances. The information can be the image ID, instance type, key pairs, security groups, and block device mapping. To learn more about Amazon machine images (AMI), see AMI Basics.

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

Auto Scaling starts by launching the minimum number (or the desired number, if specified) of EC2 instances and then starts executing the scaling plan.

If your scaling plan includes dynamic scaling then when the conditions that you specified are met, the Auto Scaling group will either scale out by launching additional EC2 instances until the maximum number of specified instances is reached, or the Auto Scaling group will scale in by terminating EC2 instances until the number of running EC2 instances equals the minimum number of specified instances.

If your scaling plan includes scaling based on a schedule, then, depending on the scaling plan, Auto Scaling will either scale out by launching additional instances or scale in by terminating instances at the scheduled time.

During normal traffic loads the Auto Scaling group maintains the number of running instances at the minimum number (or desired number, if specified) that you defined.

Auto Scaling also supports maintaining the minimum number (or the desired number, if specified) of running EC2 instances at all times without associating a scaling plan with your Auto Scaling group. You can manually adjust the number of running instances in your Auto Scaling group at any time.

Auto Scaling helps you make efficient use of your compute resources by automatically doing the work of scaling for you. This automatic scaling is the core value of the Auto Scaling service.

In addition to launching and terminating EC2 instances on demand, Auto Scaling also ensures that the EC2 instances within the Auto Scaling group are running and in good shape. Auto Scaling performs a periodic health check on current instances within an Auto Scaling group, and when it finds an unhealthy instance, it terminates that instance and launches a new one. This helps in maintaining the number of running instances at the minimum number (or desired number, if specified) that you defined.

Concepts

Topics

This section explores Auto Scaling concepts and terminology briefly introduced in the How Auto Scaling Works (p. 2) section. For information on creating your own Auto Scaling process using these concepts, see Basic Scenario in Auto Scaling.

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. You create Auto Scaling groups by defining the minimum, maximum, and desired number of running EC2 instances the group must have at any given point of time.

An Auto Scaling group starts by launching the minimum number (or the desired number, if specified) of EC2 instances and then increases or decreases the number of running EC2 instances automatically according to the conditions that you define. Auto Scaling also maintains the current instance levels by conducting periodic health check on all the instances within the Auto Scaling group. If an EC2 instance within the Auto Scaling group becomes unhealthy, Auto Scaling terminates the unhealthy instance and launches a new one to replace the unhealthy instance. This automatic scaling and maintenance of the instance levels in an Auto Scaling group is the core value of the Auto Scaling service. For information on creating an Auto Scaling group, see Basic Auto Scaling Configuration (p. 32).

Launch Configuration

A *launch configuration* is a template that the Auto Scaling group uses to launch Amazon EC2 instances. You create the launch configuration by including information such as the Amazon machine image ID to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings. When you create your Auto Scaling group, you must associate it with a launch configuration. You can attach only one launch configuration to an Auto Scaling group at a time. Launch configurations cannot be modified. They are immutable. If you want to change the launch configuration of your Auto Scaling group, you have to first create a new launch configuration and then update your Auto Scaling group by attaching the new launch configuration. When you attach a new launch configuration to your Auto Scaling group, any new instances will be launched using the new configuration parameters. Existing instances are not affected. For information on creating launch configuration, see Basic Auto Scaling Configuration (p. 32).

Amazon CloudWatch Alarms

An Amazon CloudWatch *alarm* is an object that monitors a single metric over a specific time period. A metric is a variable that you want to monitor, such as average CPU usage of the Amazon EC2 instances, or incoming network traffic from many different Amazon EC2 instances. The alarm changes its state when the value of the metric breaches a defined range and maintains the change for a specified number of periods.

An alarm has three possible states:

- `OK`—This is the state the alarm is in when the value of the metric remains within the range you've specified.
- `ALARM`—This is the state the alarm goes to when the value of the metric goes out of the range you've specified and remains outside of the range for a specified time duration.
- `INSUFFICIENT_DATA`—When the alarm is in this state, it either means that the metric is not yet available or not enough data is available for the metric to determine the alarm state.

When the alarm changes to ALARM state and remains in that state for a number of time periods, it invokes one or more actions. The actions can be a message sent to an Auto Scaling group to change the desired capacity of the group.

You configure an alarm by identifying the metrics to monitor. For example, you can configure an alarm to watch over the average CPU usage of the EC2 instances in an Auto Scaling group.

You'll have to use Amazon CloudWatch to identify metrics and create alarms. For more information, see Creating Amazon CloudWatch Alarms in the *Amazon CloudWatch Developer Guide*.

Auto Scaling Policy

An *Auto Scaling policy* is a set of instructions for Auto Scaling that tells the service how to respond to Amazon CloudWatch alarm messages. The Auto Scaling policy can give instructions to scale in (terminate EC2 instances) or scale out (launch EC2 instances) the Auto Scaling group.

In addition to creating a launch configuration, an Auto Scaling group, and a CloudWatch alarm, you'll need to create scaling in and scaling out policies and associate the policies with the Auto Scaling group.

For example, you can configure an alarm to monitor the CPU usages of the EC2 instances in your Auto Scaling Group, and you can create and associate a policy for scaling out. This scale-out policy can state that when the average usage is at 80%, then launch 20 new EC2 instances. Based on the same metric CPU usage, you can create and associate a second policy for scaling in. This scale-in policy can state that if the average CPU usage of the EC2 instances in the Auto Scaling Group falls down to 40%, then terminate 20 EC2 instances.

When the metrics (CPU usage) breaches the defined thresholds (80 % for scale-out or 40% for scale-in), the CloudWatch alarm sends a message to the associated Auto Scaling policy. Auto Scaling then executes the associated policy on the Auto Scaling group. For more information on Auto Scaling policies, see Scale Based on Demand (p. 48).

Availability Zones and Regions

Amazon cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data centers are in several physical locations. These locations are categorized by Regions and Availability Zones. Regions are large and widely dispersed geographic locations. Availability Zones are distinct locations within a region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. For information about this product's regions and endpoints, go to Regions and Endpoints in the *Amazon Web Services General Reference*.

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

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

Instance Distribution and Balance Across Multiple Zones

Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group. Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Auto Scaling will attempt to launch in other zones until it succeeds.

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

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

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

Multi-Zone Instance Counts When Approaching Capacity

Because Auto Scaling always attempts to launch new instances before terminating old ones when attempting to balance across multiple zones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities. To avoid this problem, the system can temporarily exceed the specified maximum capacity of a group by a 10 percent margin (or by a 1-instance margin, whichever is greater) during a rebalancing activity. The margin is extended only if the group is at or near maximum capacity and needs rebalancing, either 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.

Load Balancing

You can optionally use a load balancer to distribute traffic to the EC2 instances in your Auto Scaling group. A load balancer distributes incoming traffic across multiple instances in your Auto Scaling group in a way that minimizes the risk of overloading one single instance. Auto Scaling supports the use of Elastic Load Balancing load balancers. You can use Elastic Load Balancing to create a load balancer and then register your Auto Scaling group with the load balancer. After you've created your load balancer and registered your Auto Scaling group with the load balancer, your load balancer acts as a single point of contact for all incoming traffic. You can associate multiple load balancers with a single Auto Scaling group. You can also configure your Auto Scaling group to use Elastic Load Balancing metrics (such as request latency or request count) to scale your application. To learn more about creating and managing an Elastic Load Balancing load balancer, see Get Started with Elastic Load Balancing in the *Elastic Load Balancing Developer Guide*. For information on attaching a load balancer to your Auto Scaling group, see Load Balance Your Auto Scaling Group (p. 126).

Health Check

Auto Scaling periodically performs health checks on the instances in your group and replaces instances that fail these checks. By default, these health checks use the results of Amazon EC2 instance status checks to determine the health of an instance. If you use a load balancer with your Auto Scaling group, you can optionally choose to include the results of Elastic Load Balancing health checks.

Auto Scaling marks an instance unhealthy if the calls to the Amazon EC2 action DescribeInstanceStatus returns any other state other than `running`, the system status shows `impaired`, or the calls to Elastic Load Balancing action DescribeInstanceHealth returns `OutOfService` in the instance state field.

After an instance is marked unhealthy as a result of an Amazon EC2 or Elastic Load Balancing health check, it is almost immediately scheduled for replacement.

You can customize the health check conducted by your Auto Scaling group by specifying additional checks or by having your own health check system and then sending the instance's health information directly from your system to Auto Scaling.

For more information on Auto Scaling health check, see Maintain a Fixed Number of Running EC2 Instances (p. 43).

For information on adding Elastic load Balancing health check, see Add an Elastic Load Balancing Health Check to your Auto Scaling Group (p. 133). For information on adding a customized health check, see Configure Health Checks for Your Auto Scaling Group (p. 107).

To learn more about Amazon EC2 status checks, see Monitoring the Status of your Instances in the *Amazon Elastic Compute Cloud User Guide*. To learn more about Elastic Load Balancing healthchecks, see Elastic Load Balancing Health Check in the *Elastic Load Balancing Developer Guide*.

Instance LifeCycle State

The Amazon EC2 instances within your Auto Scaling group progresses through the following states over their lifespan.

- `Pending`— refers to the state when the instance is in the process of launching.
- `InService`— refers to the state when the instance is live and running.
- `Terminating`— refers to the state when the instance is in the process of being terminated.
- `Terminated`— refers to the state when the instance is no longer in service. Auto Scaling removes the terminated instance from the Auto Scaling group as soon as it is terminated. This state is not currently used.
- `Quarantined`— refers to a state that is currently not used.

You can use the DescribeAutoScalingInstances action or the `as-describe-auto-scaling-instances` command to see the lifecycle state of your instance.

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. Auto Scaling can invoke a scaling activity to rebalance an Availability Zone, to maintain the desired capacity of an Auto Scaling group, or to perform any other long-running operation supported by the service. You can use the DescribeScalingActivities action or the `as-describe-scaling-activities` command to see the scaling activities invoked by your Auto Scaling group.

Cooldown Period

A *default cooldown* period, when specified, indicates the amount of time after a scaling activity completes before any other scaling activity can start. The default cooldown period is associated with your Auto Scaling group and can be specified when creating or updating your Auto Scaling group. If a default cool down period is not specified for the Auto Scaling group, Auto Scaling uses the default value of 300 as the default cool down period for the group. For more information, see CreateAutoScalingGroup.

A *cooldown* period, when specified, indicates the amount of time during which Auto Scaling does not allow the desired size of the Auto Scaling group to be changed by any other notification from a CloudWatch alarm. A cooldown period gives the system time to perform and adjust to the most recent scaling activities (such as scale-in and scale-out) that affect capacity. This period also allows the effect of a scaling activity to become visible in the metrics that originally triggered the activity. The cool down period is associated with the Auto Scaling policy and can be specified when creating or updating an Auto Scaling policy. Use the `cooldown` option to specify a different cooldown period than the `DefaultCooldown` period specified in the Auto Scaling group. For more information, see PutScalingPolicy.

When specified, the cool down period associated with your Auto Scaling group takes priority over the default cool down period specified in the Auto Scaling group. If the policy does not specify a cool down period, the group's default cool down period is used.

You can choose to initiate a scaling activity that ignores the cool down period. When you chose this option, you can circumvent the restriction of the cool down period and change the size of the Auto Scaling group before the cool down period ends. For more information, see SetDesiredCapacity action.

Auto Scaling Instance Termination

Auto Scaling launches and terminates Amazon EC2 instances automatically in response to a scaling activity or to replace an unhealthy instance. A scaling activity can be invoked to rebalance an Availability Zone, to maintain the desired capacity of an Auto Scaling group, or to perform any other long-running operation supported by the service.

Auto Scaling uses the launch configuration associated with your Auto Scaling group to launch instances. Auto Scaling uses a termination policy, which is a set of criteria used for selecting an instance to terminate, when it must terminate one or more instances. By default, Auto Scaling uses the default termination policy, but you can opt to specify a termination policy of your own.

Before Auto Scaling selects an instance to terminate, it first identifies the Availability Zone that has more instances than the other Availability Zones used by the group. If all Availability Zones have the same number of instances, it identifies a random Availability Zone. Within the identified Availability Zone, Auto Scaling uses the termination policy to select the instance for termination.

For more information on the Auto Scaling termination policies, go to Instance Termination Policy for Your Auto Scaling Group .

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, it starts the process for terminating the instance.

Scaling Plans

Auto Scaling provides you with the following ways to configure your Auto Scaling group:

Maintain current instance levels at all times

You can configure your Auto Scaling group to maintain a minimum number (or a desired number, if specified) of running instances at all times. To maintain the current instance levels, Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. And when it finds that an instance is unhealthy, it terminates that instance and launches a new one. For more information on configuring your Auto Scaling group to maintain the current instance levels, see Maintain a Fixed Number of Running EC2 Instances (p. 43).

Manual scaling

Manual scaling is the most basic way to scale your resources. You only need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity. For more information on manually scaling your Auto Scaling group, see Change the Size of Your Auto Scaling Group (p. 45).

Scale based on a schedule

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

actions are performed automatically as a function of time and date. For more information on configuring your Auto Scaling group to scale based on a schedule, see Scale Based on a Schedule (p. 65).

Scale based on demand

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

Note that you should have two policies, one for scaling in (terminating instances) and one for scaling out (launching instances), for each event that you want to monitor. For example, if you want to scale out when the network bandwidth reaches a certain level, you'll create a policy telling Auto Scaling to start a certain number of instances to help with your traffic. But you also want an accompanying policy to scale in by a certain number when the network bandwidth level goes back down.For more information on configuring your Auto Scaling group to scale based on demand, see Scale Based on Demand (p. 48).

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. You can suspend one or more 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 desired capacity of the Auto Scaling group 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:

If you suspend...	Auto Scaling...
Alarm notifications	Ignores all Amazon CloudWatch notifications.
Availability Zone rebalance	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.
Health check	Will not automatically check instance health. Auto Scaling will still replace instances that is marked as unhealthy.
Launch	Does not launch new instances for any reason. Suspending the launch process effectively suspends the Availability Zone rebalance and replace unhealthy instance processes.
Replacing unhealthy instance	Does not replace instances marked as unhealthy. Auto Scaling continues to automatically mark instances as unhealthy.
Scheduled actions	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.

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
Auto Scaling allows you to resume both, suspended and an administrative process.

To learn more about suspending and then resuming scaling processes for your Auto Scaling group, see Suspend and Resume Auto Scaling Process (p. 114).

Tagging

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

For more information about using tags with Auto Scaling groups, go to Tag Your Auto Scaling Groups and Amazon EC2 Instances (p. 93).

Where Do I Go From Here?

After you have read What is Auto Scaling? (p. 1) and you have decided to use Auto Scaling to scale your Amazon EC2 instances, it's time to get familiar with Auto Scaling and explore how you can make use of the many features of Auto Scaling to provide solutions for your use case.

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

How Do I...	Relevant Resources
Learn more about the business case for Auto Scaling	Auto Scaling product information
Learn how Auto Scaling works	What is Auto Scaling? (p. 1)
Learn about different interfaces I can use to interact with Auto Scaling	Get Started With Auto Scaling Interfaces (p. 12) • Install the Command Line Interface (p. 14) • Use Query Requests to Call Auto Scaling APIs (p. 20) • Use the AWS SDKs (p. 28) • Manage Your AWS Credentials (p. 29)
Build the basic infrastructure that will get Auto Scaling started for most applications	Basic Auto Scaling Configuration (p. 32)

How Do I...	Relevant Resources
Manage Auto Scaling groups	Managing Your Auto Scaling Groups (p. 42) • Configure a Scaling Plan (p. 43) • Configure Instance Termination Policy for Your Auto Scaling Group (p. 75) • Launch Auto Scaling Instances into Amazon VPC (p. 100) • Configure Health Checks for Your Auto Scaling Group (p. 107) • Tag Your Auto Scaling Groups and Amazon EC2 Instances (p. 93) • Merge Your Auto Scaling Groups into a Single Multi-Zone Group (p. 110) • Suspend and Resume Auto Scaling Process (p. 114) • Shut Down Your Auto Scaling Process (p. 117)
Learn about different ways I can use Auto Scaling with other AWS services to provide solutions for my use cases	Working With Auto Scaling (p. 125) • Load Balance Your Auto Scaling Group (p. 126) • Use Amazon SQS Queues to Determine When to Auto Scale (p. 140) • Get Email Notifications When Your Auto Scaling Group Changes (p. 156) • Launch Spot Instances in Your Auto Scaling Group (p. 161) • Control Access to Auto Scaling Resources (p. 173) • Launch Auto Scaling Instances with IAM Role (p. 176) • Monitor Your Auto Scaling Instances (p. 179)
Troubleshoot Auto Scaling	Troubleshooting Auto Scaling (p. 183)
Get the technical FAQ	Auto Scaling Technical FAQ
Get help from the community of developers	Amazon EC2 Discussion Forums

Get Started With Auto Scaling Interfaces

Topics

You can create, access, and manage your Auto Scaling groups using any one of the following Amazon Web Services (AWS) Auto Scaling interfaces: the Query API, the SDK for Auto Scaling, or the command line interface (CLI).

Auto Scaling Query API—Auto Scaling provides APIs that you can call by submitting a Query Request. Query requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named *Action*. Calling an API using a Query request is the most direct way to access a web service, but requires that your application handle low-level details such as generating the hash to sign the request, and error handling. The benefit of using a Query request is that you have access to the complete functionality of an API. For information on using the Query API, see Use Query Requests to Call Auto Scaling APIs (p. 20).

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

A disadvantage of using the SDKs is that the implementation of the wrapper functions sometimes lags behind changes to the web service's API, meaning that there may be a period between the time that a new web service API is released and when a wrapper function for it becomes available in the SDKs. You can overcome this disadvantage by using the SDKs to generate a raw Query request. For more information on downloading and using the AWS SDKs, see Use the AWS SDKs (p. 28).

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

no installation required, simply download and unzip it. For more information on installing and using the Auto Scaling command line interface, see Install the Command Line Interface (p. 14).

You'll need AWS credentials to access the Auto Scaling CLI, the Auto Scaling Query API, and the Auto Scaling SDK. For information on creating and managing your AWS credentials, see Manage Your AWS Credentials (p. 29).

Install the Command Line Interface

Topics

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

An installation of a Java version 5 or later –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 interface 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 Linux/UNIX example sets JAVA_HOME for a Java executable in the `/usr/local/jre/bin` directory.

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

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

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

Note
Don't include the bin directory in JAVA_HOME; that's a common mistake some users make. The command line interface won't work if you do.

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

```
$ $JAVA_HOME/bin/java -version
java version "1.6.0_33"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.6.0_33-b03)
Java HotSpot(TM) Client VM (build 1.6.0_33-b03, mixed mode, sharing)
```

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

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.6.0_33"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.6.0_33-b03)
Java HotSpot(TM) Client VM (build 1.6.0_33-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 and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it.

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

How to Tell the Tools Where They Live

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

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

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

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

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

In addition, 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
C:\> setx PATH %PATH%;%AWS_AUTO_SCALING_HOME%\bin
```

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

How to Tell the Tools Who You Are

You must also provide your AWS credentials to the command line tool. The command line tool reads your credentials from a credential file that you create on your local system.

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

To set up security credentials for your command line tool

1. Log in to the AWS security credentials website.
2. Retrieve an access key and its corresponding secret key.
   a. Scroll down to the **Access Credentials** section and select the **Access Keys** tab.
   b. Locate an active Access Key in the **Your Access Keys** list.
   c. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.
   d. Write down the keys or save them.
   e. If no Access Keys appear in the list, click **Create a New Access Key** and follow the on-screen prompts.
3. Add your access key ID and secret access key to the file named `credential-file-path.template`:
   a. Open the file `credential-file-path.template` included in your command line interface (CLI) archive.
   b. Copy and paste your access key ID and secret access key into the file.
   c. Rename the file and save it to a convenient location on your computer.
   d. If you are using Linux, set the file permissions as follows:

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

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

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

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

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

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

How to Change the Region

By default, the Auto Scaling command line interface (CLI) uses the US East Region (us-east-1) with the autoscaling.us-east-1.amazonaws.com service endpoint URL. If your instances are in a different region, you must specify the region where your instances reside. For example, if your instances are in Europe, you must specify the eu-west-1 region by using the --region eu-west-1 parameter or by setting the AWS_AUTO_SCALING_URL environment variable.

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

To specify a different Region

1. View available Regions by going to Regions and Endpoints in the *Amazon Web Services General Reference*.
2. If you want to change the service endpoint, set the AWS_AUTO_SCALING_URL environment variable as follows:

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

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

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

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

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

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

Verify if Auto Scaling Command Line Interface is Installed

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

To verify your Auto Scaling installation and configuration

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

```
Command Name                                  Description

------------                                  -----------

as-create-auto-scaling-group                Create a new Auto Scaling group.
as-create-launch-config                      Creates a new launch configura
tion.
as-create-or-update-tags                     Create or update tags.
as-delete-auto-scaling-group                 Deletes the specified Auto
Scaling group.
as-delete-launch-config                      Deletes the specified launch
configuration.
as-delete-notification-configuration       Deletes the specified notification
 configuration.
as-delete-policy                             Deletes the specified policy.
as-delete-scheduled-action                  Deletes the specified scheduled
 action.
as-delete-tags                               Delete the specified tags
as-describe-adjustment-types                Describes all policy adjustment
 types.
as-describe-auto-scaling-groups              Describes the specified Auto
Scaling groups.
as-describe-auto-scaling-instances           Describes the specified Auto
Scaling instances.
as-describe-auto-scaling-notification-types Describes all Auto Scaling noti
fication types.
as-describe-launch-configs                   Describes the specified launch
 configurations.
as-describe-metric-collection-types          Describes all metric colle...
metric granularity types.
as-describe-notification-configurations     Describes all notification...given
 Auto Scaling groups.
as-describe-policies                        Describes the specified policies.
as-describe-process-types                  Describes all Auto Scaling process
 types.
as-describe-scaling-activities              Describes a set of activit...ties
 belonging to a group.
as-describe-scheduled-actions              Describes the specified scheduled
 actions.
as-describe-tags                             Describes tags
as-describe-termination-policy-types        Describes all Auto Scaling ter
mination policy types.
as-disable-metrics-collection                Disables collection of Auto
Scaling group metrics.
```

```
as-enable-metrics-collection                Enables collection of Auto
Scaling group metrics.
as-execute-policy                           Executes the specified policy.
as-put-notification-configuration          Creates or replaces notifi...or
 the Auto Scaling group.
as-put-scaling-policy                       Creates or updates an Auto
Scaling policy.
as-put-scheduled-update-group-action        Creates or updates a scheduled
 update group action.
as-resume-processes                         Resumes all suspended Auto...
given Auto Scaling group.
as-set-desired-capacity                     Sets the desired capacity of
the Auto Scaling group.
as-set-instance-health                     Sets the health of the instance.
as-suspend-processes                        Suspends all Auto Scaling ...
given Auto Scaling group.
as-terminate-instance-in-auto-scaling-group Terminates a given instance.
as-update-auto-scaling-group                Updates the specified Auto
Scaling group.
help
version                                     Prints the version of the CLI
tool and the API.

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

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

Use Query Requests to Call Auto Scaling APIs

Topics

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

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

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

Signing Query Requests

Query requests travel over the Internet using either HTTP or HTTPS, and are vulnerable to being intercepted and altered in transit. To prevent this and ensure that the incoming request is both from a valid AWS account and unaltered, AWS requires all requests to be signed.

To sign a Query request, you calculate a digital signature using a cryptographic hash function over the text of the request and your AWS secret key. A cryptographic hash is a one-way function that returns unique results based on the input.

When Auto Scaling receives the request, it re-calculates the signature using the request text and the secret key that matches the AWS access key in the request. If the two signatures match, Auto Scaling knows that the query has not been altered and that the request originated from your account. This is one reason why it is important to safeguard your private key. Any malicious user who obtains it would be able to make AWS calls, and incur charges, on your account.

For additional security, you should transmit your requests using Secure Sockets Layer (SSL) by using HTTPS. SSL encrypts the transmission, protecting your request or the response from being viewed in transit. For more information about securing your Query requests, see Making Secure Requests to Amazon Web Services.

The signature format that AWS uses has been refined over time to increase security and ease of use. Auto Scaling supports Signature Version 2 and Signature Version 4. If you are creating new applications that use Auto Scaling, then we recommend using Signature Version 4 for signing your query requests.

For information on how to create the signature using Signature Version 4, see Signature Version 4 Signing Process in the *AWS General Reference*.

The following sections describe the steps needed to create a query request using Signature Version 2.

Structure of an Auto Scaling Query Request

Auto Scaling requires that each HTTP or HTTPS query request formatted for Signature Version 2 contain the following:

- **Endpoint—**Also known as the host part of an HTTP request. This is the DNS name of the computer to which you send the Query request. This is different for each AWS Region. For the complete list of endpoints supported by a web service, see Regions and Endpoints in the *Amazon Web Services General Reference*.
  The endpoint, `autoscaling.amazonaws.com`, shown in the example below, is the default endpoint and maps to the Region `us-east-1`.
- **Action—**Specifies the action that you want a web service to perform.
  This value determines the parameters that are used in the request. For descriptions of Auto Scaling actions and their parameters, see Auto Scaling API Reference.

  The action in the example below is `DescribeAutoScalingGroups`, which causes a web service to return details about one or more Auto Scaling groups.
- **Required and optional parameters—**Each action in a web service has a set of required and optional parameters that define the API call. For a list of parameters that must be included in every a web service action, see Common Parameters. For details about the parameters for a specific action, see the entry for the specific Actions in the *Auto Scaling API Reference*.

  The following common parameters are required to authenticate a query request:
  - **AccessKeyId—**A value distributed by AWS when you sign up for an AWS Account. For information on retrieving the AccessId for your AWS account, see Get Your Access Key ID and Secret Access Key
  - **Timestamp—**This is the time at which you make the request. Including this in the Query request helps prevent third parties from intercepting your request and re-submitting to a web service.
  - **SignatureVersion—**The version of the AWS signature protocol you're using. Auto Scaling supports signature version 2 and signature version 4.
  - **SignatureMethod—**The hash-based protocol you are using to calculate the signature. This can be either HMAC-SHA1 or HMAC-SHA256 for version 2 AWS signatures.
  - **Signature—**This is a calculated value that ensures the signature is valid and has not been tampered with in transit.

Process for Signing Query Request Using Signature Verion 2

Before you can sign the Query request, you must put the request into a completely unambiguous format. This is needed because there are different—and yet correct—ways to format a Query request, but the different variations would result in different HMAC signatures. Putting the request into an unambiguous, canonical, format before signing it ensures that your application and a web service will calculate the same signature for a given request.

The unambigous string to sign is built up by concatenating the Query request components together as follows. As an example, let's generate the string to sign for the following call to the `DescribeAutoScalingGroups` API.

```
https://autoscaling.amazonaws.com?Action=DescribeAutoScalingGroups
&Version=2011-01-01
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2011-10-03T15%3A19%3A30
```

To create the string to sign (Signature Version 2)

1. Start with the request method (either GET or POST), followed by a newline character. (In the following, for human readability, the newline character is represented as `\n`.)

```
GET\n
```

2. Add the HTTP host header in lowercase, followed by a newline character. The port information is omitted if it is the standard port for the protocol (port 80 for HTTP and port 443 for HTTPS), but included if it is a non-standard port.

```
autoscaling.amazonaws.com\n
```

3. Add the URL-encoded version of the absolute path component of the URI (this is everything between the HTTP host header to the question mark character (?) that begins the query string parameters) followed by a newline character. If the absolute path is empty, use a forward slash (/).

```
/\n
```

4. Add the query string components (the name-value pairs, not including the initial question mark (?) as UTF-8 characters which are URL encoded per RFC 3986 (hexadecimal characters must be uppercased) and sorted using lexicographic byte ordering. Lexicographic byte ordering is case sensitive.

   Separate parameter names from their values with the equal sign character (=) (ASCII character 61), even if the value is empty. Separate pairs of parameter and values with the ampersand character (&) (ASCII code 38).

   Some API actions 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 for Auto Scaling group names looks like this:

```
&AutoScalingGroupNames.member.1=my-test-asg-1
&AutoScalingGroupNames.member.2=my-test-asg-2
```

   All reserved characters *must* be escaped. All unreserved characters *must not* be escaped. Concatenate the parameters and their values to make one long string with no spaces between them. Spaces within a parameter value, are allowed, but must be URL encoded as %20. In the concatenated string, period characters (.) are not escaped. RFC 3986 considers the period character an unreserved character, and thus it is not URL encoded.

   Note
   RFC 3986 does not specify what happens with ASCII control characters, extended UTF-8 characters, and other characters reserved by RFC 1738. Since any values may be passed into a string value, these other characters should be percent encoded as %XY where X and Y are uppercase hex characters. Extended UTF-8 characters take the form %XY%ZA... (this handles multi-bytes). The space character should be represented as '%20'. Spaces should *not* be encoded as the plus sign (+) as this will cause as error.

   The following example shows the query string components of a call to the Auto Scaling API `DescribeAutoScalingGroups`, processed as described above.

```
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Action=DescribeAutoScalingGroups&Signa
tureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2011-10-
03T15%3A19%3A30&Version=2011-01-01
```

5. The string to sign for the call to `DescribeAutoScalingGroups` takes the following form:

```
GET\n
autoscaling.amazonaws.com%0A
/\n
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Action=DescribeAutoScalingGroups&Signa
tureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2011-10-
03T15%3A19%3A30&Version=2011-01-01
```

Calculating the AWS Signature Version 2

After you've created the canonical string as described in Process for Signing Query Request Using Signature Verion 2 (p. 21), you calculate the signature by creating a hash-based message authentication code (HMAC) using either the HMAC-SHA1 or HMAC-SHA256 protocols. The HMAC-SHA256 protocol is preferred.

You then add the value returned to the Query request as a signature parameter, as shown below. You can then use the signed request in an HTTP or HTTPS call. A web service will then return the results of the call formatted as a response. For more information about the inputs and outputs of a web service API calls, see the http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/.

```
https://autoscaling.amazonaws.com?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Action=De
scribeAutoScalinggroups&SignatureMethod=HmacSHA256&SignatureVer
sion=2&Timestamp=2011-10-03T15%3A19%3A30&Version=2011-01-01&Signa
ture=lptk88A0LEP2KfwM3ima33DUjY0e%2FyfF7YfitJ%2FQw6I%3D
```

The AWS SDKs offer functions to generate Query request signatures. To see an example using the AWS SDK for Java, go to Using the Java SDK to Sign a Query Request (p. 26).

Example Query Request and Response Structures

Example Query Request Structure

In the Auto Scaling documentation, we leave the example GET requests unencoded to make them easier to read. To make the GET examples even easier to read, Auto Scaling documentation presents them in the following parsed format.

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-test-asg
&MaxRecords=20
&Action=DescribeAutoScalingGroups
&AWSAccessKeyId=AccessKeyID
&Timestamp=2009-01-28T21%3A49%3A59.000Z
&SignatureVersion=2
```

```
&SignatureMethod=HmacSHA256
&Signature=calculated value
```

The first line represents the *endpoint* of the request. This is the resource the request acts on.

After the endpoint is a question mark (?), which separates the endpoint from the parameters. Each parameter is separated by an ampersand (&).

The Action parameter indicates the action to perform (for a list of the actions, see Auto Scaling API Reference). For a list of the other parameters that are common to all Query requests, see Common Parameters in the *Auto Scaling API Reference*.

In the example Query requests we present in the Auto Scaling documentation, we omit the parameters related to authentication to make it easier to focus on the ones relevant to the particular action. We replace them with the following literal string to remind you that a real request includes the parameters: `&AUTHPARAMS`.

The above example without the authentication parameters listed will look similar to the following example:

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

Example Query Response Structure

In response to a Query request, the Auto Scaling returns an XML data structure that conforms to an XML schema defined as part of the Auto Scaling WSDL. The structure of an XML response is specific to the associated request. In general, the response data types are named according to the operation performed and whether the data type is a container (can have children). Examples of containers include `<AutoScalingGroups>` (see the example that follows). Member elements are children of containers, and their contents vary according to the container's role.

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.

If the query request is successful, you'll get a response similar to the following example:

```
DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-02-12T22:14:49.235Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-6fecd61f</InstanceId>
```

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

Troubleshooting Request Signatures Version 2

This section describes some error codes you might see when you are initially developing code to generate the signature to sign Query requests.

SignatureDoesNotMatch Signing Error in a web service

The following error response is returned when a web service attempts to validate the request signature by recalculating the signature value and generates a value that does not match the signature you appended to the request. This can occur because the request was altered between the time you sent it and the time it reached a web service endpoint (this is the case the signature is designed to detect) or because the signature was calculated improperly. A common cause of the error message below is not properly creating the string to sign, such as forgetting to URL encode characters such as the colon (:) and the forward slash (/) in Amazon S3 bucket names.

```
<ErrorResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <Error>
    <Type>Sender</Type>
    <Code>SignatureDoesNotMatch</Code>
    <Message>The request signature we calculated does not match the signature
you provided.
    Check your AWS Secret Access Key and signing method.
    Consult the service documentation for details.</Message>
  </Error>
  <RequestId>7589637b-e4b0-11e0-95d9-639f87241c66</RequestId>
```

```
</ErrorResponse>
```

IncompleteSignature Signing Error in a web service

The following error indicates that signature is missing information or has been improperly formed.

```
<ErrorResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <Error>
    <Type>Sender</Type>
    <Code>IncompleteSignature</Code>
   <Message>Request must contain a signature that conforms to AWS standards</Mes
sage>
  </Error>
  <RequestId>7146d0dd-e48e-11e0-a276-bd10ea0cbb74</RequestId>
</ErrorResponse>
```

Using the Java SDK to Sign a Query Request

The following example uses the amazon.webservices.common package of the AWS SDK for Java to generate an AWS Signature Version 2 Query request signature. To do so, it creates an RFC 2104-compliant HMAC signature. For more information about HMAC, go to HMAC: Keyed-Hashing for Message Authentication.

> **Note**
> Java is used in this case as a sample implementation. You can use the programming language of your choice to implement the HMAC algorithm to sign Query requests.

```
import java.security.SignatureException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import com.amazonaws.util.*;

/**
* This class defines common routines for generating
* authentication signatures for AWS Platform requests.
*/
public class Signature {
    private static final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     * * @param data
     * The signed data.
     * @param key
     * The signing key.
     * @return
     * The Base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     * java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
```

```
    throws java.security.SignatureException
    {
        String result;
        try {

            // Get an hmac_sha256 key from the raw key bytes.
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes("UTF8"),
 HMAC_SHA256_ALGORITHM);

            // Get an hmac_sha256 Mac instance and initialize with the signing
 key.
            Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
            mac.init(signingKey);

            // Compute the hmac on input data bytes.
            byte[] rawHmac = mac.doFinal(data.getBytes("UTF8"));

            // Base64-encode the hmac by using the utiliy in the SDK
            result = BinaryUtils.toBase64(rawHmac);


        } catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " + e.get
Message());
        }
        return result;
    }
}
```

Use the AWS SDKs

Topics

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

Type of Access	Description
AWS SDKs	AWS provides the following SDKs: • AWS SDK for Java Documentation • AWS SDK for .NET Documentation • AWS SDK for PHP Documentation • AWS SDK for Ruby Documentation
Third-Party Libraries	Developers in the AWS developer community also provide their own libraries, which you can find at the following AWS developer centers: • AWS Java Developer Center • AWS PHP Developer Center • AWS Python Developer Center • AWS Ruby Developer Center • AWS Windows and .NET Developer Center

Manage Your AWS Credentials

Topics

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

- **Amazon Login and Password**—Used to sign up for Amazon EC2 and other services, view your bills, perform account-based tasks, and get many of your security credentials. Additionally, they are used by the AWS Management Console. For information, see Your Amazon Login and Password (p. 29).
- **Access Key ID** —Used to make Query and REST-based requests. Also commonly used by UI-based tools, such as ElasticFox. For more information, see Get Your Access Key ID (p. 30).
- **X.509 Certificate and Private Key**—Used to make SOAP API requests. For more information, see Create an X.509 Certificate and Private Key (p. 30).
- **Account ID**—Used to share resources with other AWS accounts. For more information, see View Your Account ID (p. 31).

Your Amazon Login and Password

Topics

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

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

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

1. Go to the AWS website.
2. Click **Create an AWS Account**.

   The **Amazon Web Services Sign In** page appears.
3. Enter your e-mail address, select **I am a new user**, and click 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.

View Your AWS Access Credentials

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

To view your AWS access credentials

1. Go to the AWS web site.
2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.
3. All your access credentials associated with your AWS account is displayed on **Your Security Credentials** page. Move the mouse over each credential to see the description. Click to see the details.

Get Your Access Key ID

When you create the access keys, AWS Identity and Access Management (IAM) returns the Secret Access Key and an Access Key ID, which is a public identifier for the key. You need to save these keys and give them to the user. 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.

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

To get your Access Key ID

1. Go to the AWS web site.
2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.
3. In the **Your Security Credentials** page, select **Access Keys**.
4. The **Access Keys** pane opens to display the details of your Access Keys.
5. To disable an access key, click **Make Inactive**. To re-enable the key, click **Make Active**. Click **Delete** to delete the key.
6. If no Access Keys appear in the list, click **Create Access Key** and follow the on-screen prompts.

Create an X.509 Certificate and Private Key

The X.509 Certificate and Private Key are used by the command line tools and SOAP. You can download the private key file 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.

View Your Account ID

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

To view your Account ID

1. Go to the AWS web site.
2. Point to **Your Account** and select **Security Credentials**.

   If you are not already logged in, you are prompted to do so.
3. Scroll down to the **Account Identifiers** section.
4. Locate your AWS Account ID.

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

Basic Auto Scaling Configuration

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

Auto Scaling supports launching Amazon EC2 instances within any one of the following platforms :

- **EC2-Classic** — Instances launched in EC2-Classic run in a flat network that you share with other customers. We assign each instance with a private IP address from a range of private IP addresses for the EC2-Classic network. We also assign a public IP address for your instance. For more information about Amazon EC2, see What is Amazon EC2? in the *Amazon Elastic Compute Cloud User Guide.*
- **EC2-VPC** — Instances launched in EC2-VPC run in an virtual private cloud (VPC) that is logically isolated in your AWS account. We assign each instance with a private IP address from the private IP address range of your VPC. You have complete control over the VPC assigned to you. For more information about Amazon VPC, see What is Amazon VPC in the *Amazon Virtual Private Cloud User Guide*.

  If your AWS account comes with a default virtual private cloud (default VPC), your Auto Scaling group is launched within the default VPC, by default, unless you specify a subnet from a nondefault VPC. For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

  A default VPC combines the benefits of the advanced networking features provided by Amazon VPC platform (EC2-VPC) with the ease of use of the Amazon Elastic Compute Cloud platform (EC2-Classic).

  Your default VPC automatically comes with a default subnet in each Availability Zone, an Internet Gateway connected to your default VPC, and a default security group associated with your default VPC, among other default configurations. For more information on default VPC and Subnets, see Your Default VPC and Subnets.

  When you launch your Auto Scaling group within default VPC without specifying a subnet, it is automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the Auto Scaling group into the corresponding subnet for that Availability Zone. Alternatively, you can select the Availability Zone for your Auto Scaling group by selecting its corresponding default subnet.

For information about how you can tell which platforms your AWS account supports, see Supported Platforms in the *Amazon Compute Cloud User Guide*.

If you want to create your basic Auto Scaling infrastructure within EC2-VPC , see Launch Auto Scaling Instances into Amazon VPC (p. 100).

This section walks you through the process of creating your basic Auto Scaling infrastructure within EC2-Classic or default VPC. By the end of this section, you will have:

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

To create your basic Auto Scaling infrastructure, you can use the Auto Scaling command line interface (CLI) or the Query API. If you are planning on using the CLI, be sure you have installed the tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14). For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

The following sections walk you through the steps for creating your basic infrastructure using the Auto Scaling CLI or the Query API. Follow the instructions in Using the Query API (p. 37) if you are using the Query API.

Using the Command Line Interface

Use the following Auto Scaling commands to create a launch configuration, Auto Scaling group and to verify if your Auto Scaling group is created.

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

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

Create a Launch Configuration

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

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

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

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

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

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

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

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

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

```
OK-Created launch config
```

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

Create an Auto Scaling Group

After you have defined your launch configuration, you are ready to create an Auto Scaling group.

Auto Scaling groups are the core of the Auto Scaling service. An Auto Scaling group is a collection of Amazon EC2 instances. You can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions.

To create an Auto Scaling group, use the `as-create-auto-scaling-group` CLI command. Alternatively, you can use the `CreateAutoScalingGroup` API call. For more information about the API call, go to CreateAutoScalingGroup in the *Auto Scaling API Reference*. For information about Auto Scaling groups, see Auto Scaling Group.

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

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

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

group size tells Auto Scaling the minimum and maximum number of instances the Auto Scaling group should have.

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

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

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

For this launch configuration, specify the following options:

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

Important
You will incur the standard Amazon EC2 usage fees for the instance until you terminate it as the last task in this tutorial. For more information about Amazon EC2 usage rates, go to the Amazon EC2 product page.

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

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

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

```
OK-Created AutoScalingGroup
```

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

Verify Auto Scaling Group Creation

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

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

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

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

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

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

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

Verify Auto Scaling Instances

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

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

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

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

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

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

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

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

Tasks Completed

You just performed the following tasks:

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

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

Note
The instance associated with the Auto Scaling group you just created is not launched instantly. So, if you run the snippet as a single code block, you will not see see the instance information right away.

```
as-create-launch-config my-test-lc --image-id ami-0078da69 --instance-type
m1.small
as-create-auto-scaling-group my-test-asg --launch-configuration my-test-lc --
availability-zones us-east-1a --min-size 1 --max-size 10 --desired-capacity 1
as-describe-auto-scaling-groups --headers
as-describe-auto-scaling-instances --headers
```

Using the Query API

Use the following Auto Scaling actions to create a launch configuration, Auto Scaling group and to verify if your Auto Scaling group is created.

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

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

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

Create a Launch Configuration

The launch configuration specifies the template that Auto Scaling uses to launch Amazon EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. For information about launch configuration, see Launch Configuration.

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

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

Note
The AMI ID is provided for illustration purposes only. AMI IDs change over time. You can obtain current, valid AMI IDs by calling the Amazon EC2 `DescribeImages` action.

Your request should look similar to the following example:

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

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

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

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

Create an Auto Scaling Group

After you have defined your launch configuration, you are ready to create an Auto Scaling group.

Auto Scaling groups are the core of the Auto Scaling service. An Auto Scaling group is a collection of Amazon EC2 instances. You can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions.

For information about Auto Scaling groups, see Auto Scaling Group.

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

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

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

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

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

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

- Maximum size: `10`
- Desired capacity: `1`

Important
You will incur the standard Amazon EC2 usage fees for the instance until you terminate it as the last task in this tutorial. For more information about Amazon EC2 usage rates, go to the Amazon EC2 product page.

Your request should look similar to the following example:

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

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

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

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

Verify Auto Scaling Group Creation

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

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

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

Your request should look similar to the following example:

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

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
```

```
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-02-12T22:14:49.235Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-6fecd61f</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:auto
ScalingGroup:5d1ee7f3-f0f6-42bd-851e-0513f88c56b0:autoScalingGroupName/my-test-
asg</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>10</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>e57b79d1-7564-11e2-9320-f7b1aEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Verify Auto Scaling Instances

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

Call the `DescribeAutoScalingInstances` action.

Your request should look similar to the following example:

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

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

```
<DescribeAutoScalingInstancesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingInstancesResult>
    <AutoScalingInstances>
      <member>
        <HealthStatus>HEALTHY</HealthStatus>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <AvailabilityZone>us-east-1a</AvailabilityZone>
        <InstanceId>i-6fecd61f</InstanceId>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <LifecycleState>InService</LifecycleState>
      </member>
    </AutoScalingInstances>
  </DescribeAutoScalingInstancesResult>
  <ResponseMetadata>
    <RequestId>f6e01d93-7567-11e2-90b3-8dedfEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingInstancesResponse>
```

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

Tasks Completed

You just performed the following tasks:

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

Managing Your Auto Scaling Groups

This section provides you with walkthroughs that explore different ways you can configure and manage your Auto Scaling groups. If you aren't already acquainted with the concepts behind Auto Scaling and Auto Scaling groups, see How Auto Scaling Works (p. 2). For information on basic Auto Scaling configuration, see Basic Auto Scaling Configuration (p. 32).

You can access Auto Scaling by downloading and installing the Auto Scaling command line interface (CLI), by creating a query request with the query API, or by using the Amazon SDKs. The procedures in this section include instructions for the command line interface and the Query API. For information on downloading and using the Auto Scaling interfaces, see Get Started With Auto Scaling Interfaces (p. 12).

In all our example procedures, we assume that your instances are in the US East (Northern Virginia) Region. If your instances are in a different Region, you must specify the Region where your instances reside. For example, if your instances are in Europe, you must specify the *eu-west-1* Region by using the `--region eu-west-1` parameter or setting the `EC2_REGION` environment variable. For information on specifying a Region, see How to Change the Region (p. 17).

The walkthroughs in this section lead you through the following scenarios:

- Configure a Scaling Plan (p. 43)

  This walkthrough explores in detail three different ways to configure your Auto Scaling group to scale.
- Configure Instance Termination Policy for Your Auto Scaling Group (p. 75)

  This walkthrough walks you through the process to configure termination policy for the instances in your Auto Scaling group.
- Tag Your Auto Scaling Groups and Amazon EC2 Instances (p. 93)

  Auto Scaling group tags help organize your Auto Scaling resources and provide additional information for your Auto Scaling group such as software version, role, or location information. Auto Scaling group tags also provide search, group, and filter functionality. This walkthrough walks you through the process to create tags for your Auto Scaling group.
- Launch Auto Scaling Instances into Amazon VPC (p. 100)

  This walkthrough walks you through the process to create an Auto Scaling group to launch instances within Amazon Virtual Private Cloud (VPC).
- Configure Health Checks for Your Auto Scaling Group (p. 107)

By default, Auto Scaling group determines the health state of each instance by periodically checking the results of Amazon EC2 instance status checks and Elastic Load Balancing health checks, if specified. You can configure your own health check system and set the health state of the instances in your Auto Scaling group to a state of your choice. This walkthrough walks you through the process to explicitly specify the health state of an instance in your Auto Scaling group.

- Merge Your Auto Scaling Groups into a Single Multi-Zone Group (p. 110)

  This walkthrough walks you through the processes 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.
- Suspend and Resume Auto Scaling Process (p. 114)

  This walkthrough walks you through the process to suspend and then resume all scaling activities on the Auto Scaling group.
- Shut Down Your Auto Scaling Process (p. 117)

  This walkthrough walks you through the process to completely shut down the Auto Scaling process for your Auto Scaling group.

Configure a Scaling Plan

Scaling is the ability to increase or decrease the compute capacity of your application. When you scale using Auto Scaling, capacity is automatically increased or decreased according to the conditions you define, thus ensuring that performance is maintained and the cost is minimized.

With Auto Scaling, you can plan to configure your Auto Scaling Group (p. 4) to automatically scale or maintain your application. You have three options. You can create one of the option plans:

- Maintain a Fixed Number of Running EC2 Instances (p. 43)

  Use this scaling plan if you would like Auto Scaling to maintain the minimum (or the desired, if specified) number of instances in your Auto Scaling group at all times. You can manually change the number of running instances in your Auto Scaling group at any time.
- Scale Based on Demand (p. 48)

  Use this scaling plan if you need to scale dynamically in response to changes in the demand for your application. When you scale based on demand, you must specify when and how to scale.
- Scale Based on a Schedule (p. 65)

  Use this scaling plan if you want to scale your application on a pre-defined schedule. You can specify the schedule for scaling one time only or provide details for scaling on a recurring schedule.

The following sections explore in detail these three different ways to configure your Auto Scaling group. If you aren't already acquainted with the basic concepts behind Auto Scaling, take a quick look at the overview topic: What is Auto Scaling? (p. 1)

Maintain a Fixed Number of Running EC2 Instances

Topics

- Change the Size of Your Auto Scaling Group (p. 45)

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

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

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

By default, the Auto Scaling group determines the health state of each instance by periodically checking the results of Amazon EC2 instance status checks. If the instance status description shows any other state other than `running` or if the system status description shows `impaired`, Auto Scaling considers the instance to be unhealthy and launches a replacement.

If you have associated your Auto Scaling group with a load balancer and have chosen to use the Elastic Load Balancing health check, Auto Scaling determines the health status of the instances by checking the results of both Amazon EC2 instance status and Elastic Load Balancing instance health. Auto Scaling marks an instance unhealthy if the calls to the Amazon EC2 action DescribeInstanceStatus returns any other state other than `running`, the system status shows `impaired`, or the calls to Elastic Load Balancing action DescribeInstanceHealth returns `OutOfService` in the instance state field.

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

To learn more about Amazon EC2 instance status checks, see Monitoring the Status of Your Instances in the *Amazon Elastic Compute Cloud User Guide*. To learn more about Elastic Load Balancing health checks, see Elastic Load Balancing Health Check in the *Elastic Load Balancing Developer Guide*.

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

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

The Basic Scenario in Auto Scaling shows you how to do the following activities:

1. Create a launch configuration (for example, `my-test-lc`).
2. Use the launch configuration to create an Auto Scaling group by specifying the minimum, maximum, and (optionally) the desired number of running instances you want the Auto Scaling group to have at any point of time.
3. Verify your Auto Scaling group and instances
4. Verify the health state of the instances in the Auto Scaling group

Change the Size of Your Auto Scaling Group

At any time you can change the size of an Auto Scaling group that you have configured to maintain a fixed number of running EC2 instances. You only need to specify a change to the maximum, minimum, or desired capacity of your Auto Scaling group. Auto Scaling manages the process of creating or terminating instances to maintain the updated group size.

You can change the size of your Auto Scaling group using either the Auto Scaling command line interface (CLI) or the Query API.

Using the Command Line Interface

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

```
as-set-desired-capacity my-test-asg  --desired-capacity 2 --honor-cooldown
```

By default the command overrides any cool down period specified for the Auto Scaling group. You can choose to reject the default behavior and honor the cool down period by specifying the `--honor-cooldown` option with the command. For more information on the cool down period, see Cooldown Period (p. 7).

Auto Scaling returns the following response:

```
OK-Desired Capacity Set
```

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

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

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

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

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

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

Using the Query API

Use the SetDesiredCapacity action with the following parameters to change the size of your Auto Scaling group,

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

By default the command overrides any cooldown period specified for the Auto Scaling group. Set `HonorCoolDown`to `True` if you want Auto Scaling to reject the default behavior and honor the cooldown period. For more information, see Cooldown Period (p. 7)

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

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

Use the DescribeAutoScalingGroups action with the following parameter to confirm that the size of your Auto Scaling group has changed.

- AutoScalingGroupName = `my-test-asg`

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-28T22:14:05.886Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-42b66e32</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-88ae76f8</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
```

```
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-1:123456789012:auto
ScalingGroup:fec2667a-0410-419e-a6fa-16f37Example:
        autoScalingGroupName/my-test-asg2</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>e79c8299-699a-11e2-b287-b79f0EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

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

Scale Based on Demand

Topics

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

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

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

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

You use Auto Scaling to create your scaling policies. When a scaling policy is executed, it changes the current size of your Auto Scaling group by the amount you specify in the policy. You can express the change to the current size as an absolute number, an increment, or as a percentage of the current group size. When the policy is executed, Auto Scaling uses both the current group capacity and the change specified in the policy to compute a new size for your Auto Scaling group. Auto Scaling then updates the current size, and this consequently affects the size of your group.

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

Each Auto Scaling group can have up to 25 policies.

To create a scaling policy you need to specify the policy name, the name of the Auto Scaling group to associate the policy with, and the following parameters:

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

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

Auto Scaling supports the following adjustment types:

- **ChangeInCapacity:** Use this to increase or decrease existing capacity. For example, let's say that the current capacity of your Auto Scaling group is set to three instances. You then create a scaling policy on your Auto Scaling group, specify the type as `ChangeInCapacity` and the adjustment as five. When the policy is executed, Auto Scaling will add five more instances to your Auto Scaling group. You'll then

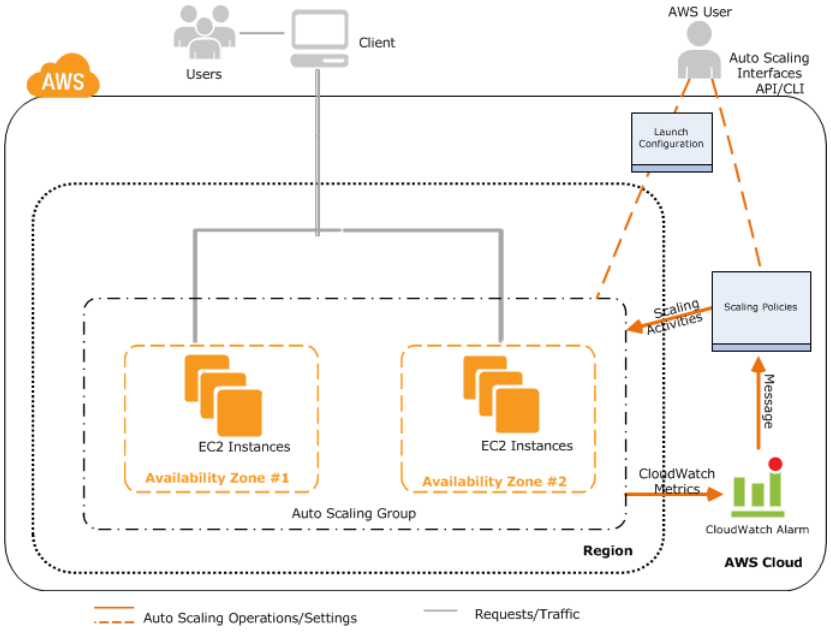
have eight running instances in your Auto Scaling group: current capacity (3) plus ChangeInCapacity (5) equals 8.

- **ExactCapacity:** Use this to change the current capacity of your Auto Scaling group to the exact value specified. For example, let's say that the capacity of your Auto Scaling group is set to five instances. You then create a scaling policy on your Auto Scaling group,specify the type as `ExactCapacity` and the adjustment as three. When the policy is executed, your Auto Scaling group will have three running instances.

  You'll get an error if you specify a negative adjustment value for the `ExactCapacity` adjustment type.
- **PercentChangeInCapacity:** Use this to increase or decrease the desired capacity by a percentage of the desired capacity. For example, let's say that the desired capacity of your Auto Scaling group is set to ten instances. You then create a scaling policy on your Auto Scaling group, specify the type as `PercentChangeInCapacity` and the adjustment as ten. When the policy is executed, your Auto Scaling group will have eleven running instances because 10 percent of 10 instances is 1 instance, and 1 instance plus 10 instances equals 11 instances.

  If `PercentChangeInCapacity` returns a value between 0 and 1, Auto Scaling will round it off to 1. If the `PercentChangeInCapacity` returns a value greater than 1, Auto Scaling will round it off to the lower value. For example, if `PercentChangeInCapacity` returns 12.5, then Auto Scaling will round it off to 12.

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



This section provides a quick walkthrough of the flow of events illustrated in the architectural diagram. These events begin when a client sends a request to an AWS user's application.

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

- Created a launch configuration by providing all the information required to launch EC2 instances.
- Created an Auto Scaling group by defining maximum, minimum, and (optionally), the desired capacity for the EC2 instances.
- Created an Amazon CloudWatch alarm and defined which metrics to monitor.

- Created two scaling policies, one for scaling out and another for scaling in, and associated the policies with the alarm.
- Associated the scaling policies with the Auto Scaling group.

The following walkthrough begins with the launch of EC2 instances in your Auto Scaling group.

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

Creating Policies and Alarms

Topics

This section walks you through creating Auto Scaling policies and Amazon CloudWatch alarms. You create CloudWatch alarms that watch over the scale-in and scale-out metrics that you specified when you created your policy. Then you associate the alarms with the scaling policies that you have created. These alarms send messages to Auto Scaling when the specified metrics breach the thresholds that you specified in your policies

The following steps outline how to create policies and alarms.

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

To create scaling policies, you can use the Auto Scaling command line interface (CLI) or the Query API. If you are planning on using the CLI, be sure you have installed the tools. For more information, see Install the Command Line Interface (p. 14).

To create CloudWatch alams, you can use the CloudWatch console, the CloudWatch CLI, or the CloudWatch Query API. If you use the CloudWatch CLI, be sure you have installed the tools. For more information, see Command Line Tools.

Using the Command Line Interface

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

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

For detailed information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the Auto Scaling Quick Reference Card.

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

Command	Description
`mon-put-metric-alarm`	Creates an alarm and associates it with the specified CloudWatch metric and the Auto Scaling policy.
`mon-describe-alarms`	Retrieves the details of the specified alarm. If no alarm is specified, retrieves details of all the alarms for the AWS account.

For detailed information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the Amazon CloudWatch Command Line Interface Reference.

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

Create Launch Configuration

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

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

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

  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- Instance type = `m1.small`

Your command should look similar to the following example:

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

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

```
OK-Created launch config
```

Create an Auto Scaling Group

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

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

Your command should look like the following example:

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

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

```
OK-Created AutoScalingGroup
```

Verify Your Auto Scaling Group

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

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

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

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

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

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

Create Scaling Policies

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

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

To create scaling policies

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

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

   Your command should look similar to the following example:

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

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

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

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

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

3. Use the `as-put-scaling-policy` command to create another policy by specifying the following values:

   - Policy name = `my-scalein-policy`
   - Auto Scaling group name = `my-test-asg`
   - Adjustment = `-2`
   - Adjustment type = `ChangeInCapacity`

   Your command should look similar to the following example:

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

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

4. Auto Scaling returns the ARN for the policy.

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

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

Scaling policies also enable you to specify a custom cooldown period. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. Because scaling activities are suspended when an Auto Scaling group is in cool down mode, an adequate cool down period helps to prevent initiating a scaling activity based on stale metrics. By default, Auto Scaling uses a default period associated with your Auto Scaling group. To use a different cooldown period than the default specified in the Auto Scaling group, use the `--cooldown` *`cooldown value`* option with the `as-put-scaling-policy` command. When specified, the policy cooldown period takes priority over the default cooldown period specified in the Auto Scaling group. If the policy does not specify a cooldown period, the group's default cooldown period is used. For more information, see Cooldown Period (p. 7).

Create CloudWatch Alarms

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

Before you begin, be sure you have installed the Amazon CloudWatch command line interface. For more information, see Command Line Tools.

To create CloudWatch alarms

1. Use the CloudWatch command `mon-put-metric-alarm` to create an alarm to for increasing the size of the Auto Scaling group when the average CPU usage of all the instances goes up to 80 percent by specifying the following values:

   - Alarm name = `AddCapacity`
   - Metric name = `CPUUtilization`
   - Namespace = `"AWS/EC2"`
   - Statistic = `Average`
   - Period = `120`
   - Threshold = `80`
   - Comparison operator = `GreaterThanOrEqualToThreshold`
   - Dimensions = `"AutoScalingGroupName=my-test-asg"`
   - Evaluation periods = `2`
   - Alarm action = `arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-test-asg:policyName/my-scaleout-policy`

   For more information about the options used in `mon-put-metric-alarm` command, see the mon-put-metric-alarm Command in the *Amazon CloudWatch Developer Guide*.

   Your command should look like the following example:

```
prompt>mon-put-metric-alarm --alarm-name AddCapacity  --metric-name
CPUUtilization --namespace “AWS/EC2”
```

```
--statistic Average --period 120 --threshold 80 --comparison-operator
GreaterThanOrEqualToThreshold --dimensions “AutoScalingGroupName=my-test-
asg”
--evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScal
ingGroupName/ my-test-asg:policyName/my-scaleout-policy
```

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

```
OK-Created Alarm
```

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

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

   For more information about the options used in the `mon-put-metric-alarm` command, see mon-put-metric-alarm Command in the *Amazon CloudWatch Developer Guide*.

   Your command should look like the following example:

```
mon-put-metric-alarm --alarm-name RemoveCapacity --metric-name CPUUtilization
 --namespace “AWS/EC2”
 --statistic Average  --period 120  --threshold 40  --comparison-operator
LessThanOrEqualToThreshold  --dimensions “AutoScalingGroupName=my-test-asg”

 --evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:123456789012:scalingPolicy:4ee9e543-86b5-4121-
b53b-aa4c23b5bbcc:autoScalingGroupName/ my-test-asg:policyName/my-scalein-
policy
```

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

```
OK-Created Alarm
```

Verify Your Scaling Policies and CloudWatch Alarms

To verify your CloudWatch alarms

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

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

2. The command returns the following:

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

To verify your scaling policies

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

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

2. The command returns the following:

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

Congratulations! You've successfully created scaling policies and CloudWatch alarms. And you have associated your scaling polices with the CloudWatch alarms.

Using the Query API

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

Command	Description
CreateLaunchConfiguration	Creates a new launch configuration with specified attributes.

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

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

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

Command	Description
PutMetricAlarm	Creates an alarm and associates it with the specified CloudWatch metric and the Auto Scaling policy.
DescribeAlarms	Retrieves the details of the specified alarm. If no alarm is specified, retrieves details of all the alarms for the AWS account.

For more information about CloudWatch actions, see the Amazon CloudWatch API Reference.

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

Create Launch Configuration

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

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

  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- InstanceType = `m1.small`

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

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

Create an Auto Scaling Group

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

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

- MaxSize = 5
- MinSize = 1

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

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

Verify Your Auto Scaling Group

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

- AutoScalingGroupName = `my-test-asg`

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

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

```
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>5</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>cb35382a-64ef-11e2-a7f1-9f203EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

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

Create scaling policies

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

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

To create scaling policies

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

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

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

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

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

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

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

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

4. The response includes the ARN for the policy.

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

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

Scaling policies also enable you to specify a custom cooldown period. Cool down periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. Because scaling activities are suspended when an Auto Scaling group is in cooldown mode, an adequate cooldown period helps to prevent the initiation of a scaling activity based on stale metrics. By default, Auto Scaling uses a default cooldown period associated with your Auto Scaling group. To use a different cooldown period than the default specified in the Auto Scaling group, use the `Cooldown` *`=cooldown value`* parameter. When specified, the policy cooldown period takes priority over the default cooldown period specified in the Auto Scaling group. If the policy does not specify a cooldown period, the group's default cool down period is used. For more information, see Cooldown Period (p. 7).

Create Cloudwatch alarms

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

To create CloudWatch alarms

1. Call the CloudWatch action `PutMetricAlarm` to create an alarm to increase the size of the Auto Scaling group when the average CPU usage of all the instances goes up to 80 percent by specifying the following parameters:

   - AlarmName = `AddCapacity`
   - MetricName = `CPUUtilization`
   - Namespace = `"AWS/EC2"`
   - Statistic = `Average`
   - Period = `120`
   - Threshold = `80`
   - ComparisonOperator = `GreaterThanOrEqualToThreshold`
   - Dimensions.member.1 = `"AutoScalingGroupName=my-test-asg"`

- EvaluationPeriods = 2
- AlarmActions.member.1 = arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-test-asg:policyName/my-scaleout-policy

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

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

2. Call the CloudWatch action `PutMetricAlarm` to create an alarm to decrease the size of the Auto Scaling group when the average CPU usage of all the instances goes down to 40 percent by specifying the following parameters:

   - AlarmName = `RemoveCapacity`
   - MetricName = `CPUUtilization`
   - Namespace = `"AWS/EC2"`
   - Statistic = `Average`
   - Period = `120`
   - Threshold = `40`
   - ComparisonOperator = `LessThanOrEqualToThreshold`
   - Dimensions.member.1 = `"AutoScalingGroupName=my-test-asg"`
   - EvaluationPeriods = `2`
   - AlarmActions.member.1 = `arn:aws:autoscaling:us-east-1:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/my-test-asg:policyName/my-scalein-policy`

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

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

Verify Your Scaling Policies and CloudWatch Alarms

To verify your CloudWatch alarms

1. Call the CloudWatch action `DescribeAlarms` using the following parameters:

   - AlarmNames.member.1 = `AddCapacity`
   - AlarmNames.member.2 = `RemoveCapacity`

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

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

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

```
          </member>
        </Dimensions>
       <ComparisonOperator>LessThanOrEqualToThreshold</ComparisonOperator>

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

To verify your scaling policies

1. Call the Auto Scaling action `DescribePolicies` with the following parameter:

   - AutoScalingGroupName = `my-test-asg`

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

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

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

Scale Based on a Schedule

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

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

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

Programming Considerations for Scheduled Actions

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

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

Create Scheduled Actions

Topics

- Using the Command Line Interface (p. 66)
- Using the Query API (p. 70)

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

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

You can create scheduled actions using the Auto Scaling command line interface (CLI) or the Query API. If you are planning to use the CLI, be sure you have installed the tools. For more information, see Install the Command Line Interface (p. 14).

Using the Command Line Interface

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

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

For detailed information about the command syntax of any of the commands listed above, use the `--help` option with the command or see the Auto Scaling Quick Reference Card.

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

Create a Launch Configuration

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

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

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

  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- Instance type = `m1.small`

Your command should look similar to the following example:

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

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

```
OK-Created launch config
```

Create an Auto Scaling Group

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

- Auto Scaling group name = `my-test-asg`
- Launch configuration name = `my-test-lc`

- Availability Zone = `us-east-1e`
- Max size = `5`
- Min size = `1`

Your command should look similar to the following example:

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

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

```
OK-Created AutoScalingGroup
```

Verify Your Auto Scaling Group

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

Your command should look similar to the following example:

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

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

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

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

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

Create a Schedule for Scaling Actions

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

To schedule scaling for one time only

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

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

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

Your command should look similar to the following example:

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

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

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

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

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

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

   Your command should look similar to the following example:

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

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

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

To create scheduled actions for scaling on a recurring schedule

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

   - Name of your scheduled action = `Scaleout-schedule-year`
   - Auto Scaling group name = `my-test-asg`
   - Desired Capacity = `3`
   - Recurrence = `“30 0 1 1,6,12 0”`

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

Your command should look similar to the following example:

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

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

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

Verify that the Auto Scaling Group is Scheduled for Scaling

To verify your Auto Scaling group is schduled for scaling

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

   Your command should similar to the following example:

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

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

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

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

   Your command should look similar to the following example:

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

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

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2
011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>71382dd3-75af-4a57-9c23-988fdcb1866a</NextToken>
    <Activities>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>100</Progress>
```

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

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

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

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

Using the Query API

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

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

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

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

Create a Launch Configuration

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

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

- LaunchConfiguration name = `my-test-lc`
- ImageId = `ami-514ac838`

  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- InstanceType = `m1.small`

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

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

Create an Auto Scaling Group

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

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

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

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

Verify Your Auto Scaling Group

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

- AutoScalingGroupName = `my-test-asg`

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazonaws.com/doc/2
011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
```

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

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

Create a Schedule for Scaling Actions

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

To schedule scaling on a specific date and time

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

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

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

Auto Scaling supports the date and time expressed in "YYYY-MM-DDThh:mm:ssZ" format in UTC/GMT only.

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

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

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

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

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

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

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

To create scheduled actions for scaling on recurrence basis

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

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

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

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

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

Verify that Your Auto Scaling Group is Scheduled for Scaling

To verify your Auto Scaling group is schduled for scaling

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

   - AutoScalingGroupName = `my-test-asg`

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

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2
011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>71382dd3-75af-4a57-9c23-988fdcb1866a</NextToken>
    <Activities>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>100</Progress>
        <ActivityId>7d1a9dd2-7b53-4334-8a5f-5fa2a9731d64</ActivityId>
        <StartTime>2013-01-30T03:00:51.931Z</StartTime>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
       <Cause>At 2013-01-30T03:00:21Z a scheduled action update of AutoScal
ingGroup constraints to min: 1, max: 5, desired: 3 changing the desired ca
pacity fro
m 1 to 3.  At 2013-01-30T03:00:21Z the scheduled action ScaleOut executed.
 Setting desired capacity from 1 to 3.  At 2013-01-30T03:00:51Z an instance
 was started
 in response to a difference between desired and actual capacity, increasing
 the capacity from 1 to 3.</Cause>
        <Details>{}</Details>
       <Description>Launching a new EC2 instance: i-aacc62da</Description>

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

You can determine if instances were launched due to a scheduled action by examining the description in the `Cause` field. Activities launched as a direct result of a scheduled action will have a reference

to the specific action name in the `Cause` field of the corresponding scaling activity. The `Cause` field in the example shows the scheduled action taken for the Auto Scaling group `my-test-asg`.

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

Configure Instance Termination Policy for Your Auto Scaling Group

Topics

Auto Scaling launches and terminates Amazon EC2 instances automatically in response to a scaling activity or to replace an unhealthy instance. A scaling activity can be invoked to rebalance an Availability Zone, to maintain the desired capacity of an Auto Scaling group, or to perform any other long-running operation supported by the service.

Auto Scaling uses the launch configuration associated with your Auto Scaling group to launch instances. For information on creating launch configurations, see Basic Auto Scaling Configuration (p. 32). Auto Scaling uses a termination policy, which is a set of criteria used for selecting an instance to terminate, when it must terminate one or more instances. By default, Auto Scaling uses the default termination policy, but you can opt to specify a termination policy of your own.

Before Auto Scaling selects an instance to terminate, it first identifies the Availability Zone that has more instances than the other Availability Zones used by the group. If all Availability Zones have the same number of instances, it identifies a random Availability Zone. Within the identified Availability Zone, Auto Scaling uses the termination policy to select the instance for termination.

How the Default Termination Policy Works

The default termination policy uses the following steps to identify an instance to terminate:

1. Within the selected Availability Zone, it identifies the instance that was launched using the oldest launch configuration.

   Auto Scaling calls the DescribeLaunchConfiguration action on the instances in an Auto Scaling group. It then uses the time in the `CreatedTime` field to identify the instance with the oldest launch configuration.
2. If more than one running instance was launched using the oldest launch configuration, it identifies the instance within that subset that is closest to the next instance hour.

   By selecting the instance that is closest to the next instance hour, Auto Scaling maximizes the hours your instances run while minimizing the billable instance-hours.
3. If more than one instance is closest to the next instance hour, it selects a random instance from that subset.

How Your Termination Policy Works

Auto Scaling provides a list of termination policies for you to choose from, in case the default termination policy does not satisfy your requirements. If you specify your termination policy, Auto Scaling will use it instead of the default termination policy. If you do not specify a termination policy, Auto Scaling will use the default termination policy.

Auto Scaling provides the following termination policy options for you to choose from. You can specify one or more of these options in your termination policy.

- **OldestInstance** — Specify this if you want the oldest instance in your Auto Scaling group to be terminated.

  Auto Scaling uses the instance launch time to identify the instance that was launched first.
- **NewestInstance** — Specify this if you want the last launched instance to be terminated.

  Auto Scaling uses the instance launch time to identify the instance that was launched last.
- **OldestLaunchConfiguration** — Specify this if you want the instance launched using the oldest launch configuration to be terminated.

  Auto Scaling calls the DescribeLaunchConfiguration action on the instances in an Auto Scaling group. It then uses the time in the `CreatedTime` field to identify the instance with the oldest launch configuration.
- **ClosestToNextInstanceHour** — Specify this if you want the instance that is closest to completing the billing hour to be terminated.

  By selecting the instance that is closest to the next instance hour, you maximize the hours your instances run while minimizing the billable instance-hours.
- **Default** — Specify this if you want Auto Scaling to use the default termination policy to select instances for termination.

To configure your termination policy, you can either specify any one of the policies as a standalone policy, or you can list multiple policies in an ordered list. The policies are executed in the order they are listed.

> **Important**
> If you are listing `Default` as part of a policy list, be sure to order the list such that `Default` is listed last in the policy list.

Your Termination policy uses the following steps to identify an instance to terminate.

1. Within the selected Availability Zone, it identifies the instance that matches the specified termination policy.
2. If more than one running instance matches the specified policy, it goes to the next policy on the list. If there are no more policies specified, it uses the default termination policy to identify an instance for termination.

Configure Your Termination Policy

Topics

- Using the Command Line Interface (p. 77)
- Using the Query API (p. 83)

You can either create a new Auto Scaling group with which you can associate your termination policy or you can update an existing Auto Scaling group to use your termination policy. If you are updating your

existing Auto Scaling group, the new termination policy is executed when the next scaling activity takes place for the Auto Scaling group.

In this section, we will walk you through the process of configuring your instance termination policy for your Auto Scaling group. You'll start by creating a new launch configuration, and then you'll create a new Auto Scaling group with your termination policy. If you would prefer to update your existing Auto Scaling group instead, skip the initial steps for creating new launch configuration and Auto Scaling group.

The following table outlines the steps for configuring your termination policy. This will be followed by instructions for configuring your termination policy using either the Auto Scaling command line interface (CLI) or the Auto Scaling Query API.

Steps	Description
Step 1	1.1 Create Launch Configuration. 1.2 Create Auto Scaling group, specifing the termination policy you want to use to terminate instances. 1.3 Or, Update your Auto Scaling group by specifying the termination policy you want to use to terminate instances.
Step 2	Verify that your Auto Scaling group lanches with your termination policy.
Step 3	Test to ensure that instances in your Auto Scaling group are getting terminated as specified in your termination policy.
Step 4	Clean up.

Using the Command Line Interface

If you do not have the latest version of the Auto Scaling command line interface (CLI) installed, go to the Auto Scaling Command Line Tool page to download it now. For instructions on installing and using the Auto Scaling CLI, see Install the Command Line Interface (p. 14) in the *Auto Scaling Developer Guide*.

You will use the following Auto Scaling commands to configure your termination policy for your Auto Scaling group.

Command	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with a specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the Auto Scaling groups, if the groups exist.
`as-update-auto-scaling-group`	Updates the configuration for the specified AutoScalingGroup.
`as-delete-auto-scaling-group`	Deletes the specified Auto Scaling group, if the group has no instances and no scaling activities are in progress.

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

For detailed information about the command syntax of any of the commands listed above, use the `--help` option with the command or go to the Auto Scaling Quick Reference Card.

1. **Create or update your Auto Scaling group.**

   If you are updating your existing Auto Scaling group to use your Termination Policy, go to step 1c.

   a. **Create launch configuration.**

      If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Auto Scaling Configuration (p. 32). Use the basic scenario to get started using the infrastructure that is typically needed in Auto Scaling.

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

      - Launch configuration name = `lc-test-termination-policy`
      - Image ID = `ami-0078da69`

        If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
      - Instance type = `m1.small`

      Your command should look similar to the following example:

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

      You should get a confirmation like the following example:

```
OK-Created launch config
```

   b. **Create Auto Scaling group.**

      Create your Auto Scaling group by using `as-create-auto-scaling-group` and then specifying the launch configuration you just created. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 34).

      Specify the following values for the `as-create-auto-scaling-group` command:

      - Auto Scaling group name = `asg-test-termination-policy`
      - Launch configuration name = `lc-test-termination-policy`
      - Availability Zone = `us-east-1e`
      - Max size = 3
      - Desired capacity = 2
      - Min size = 1
      - Termination policy = `OldestInstance`

      Your command should look like the following example:

```
as-create-auto-scaling-group asg-test-termination-policy --launch-config
uration lc-test-termination-policy --availability-zones "us-east-1e" --
max-size 3 --min-size 1 –desired-capacity 2 ––termination-policies
“OldestInstance”
```

You should get confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Skip the following step and go to step 2 to verify that your Auto Scaling group has launched instances with the Termination Policy.

c. **Update Auto Scaling group.**

Update your existing Auto Scaling group by using the `as-update-auto-scaling-group` command and specifying the following values:

- Auto Scaling group name = *`your Auto Scaling group name`*
- Termination policy = `OldestInstance`

Your command should look like the following example:

```
as-update-auto-scaling-group your Auto Scaling group name --termination-
policies "OldestInstance"
```

You should get confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

2. **Verify Auto Scaling Group has launched with your termination policy.**

To confirm that Auto Scaling has launched your EC2 instances with your termination policies, use `as-describe-auto-scaling-groups` command. The command shows details about the group and instances launched.

Specify this value for the command:

- Auto Scaling group name = `asg-test-termination-policy`

Note
If you've updated your Auto Scaling group, replace the Auto Scaling group name, `asg-test-termination-policy`, with the name of your Auto Scaling group and replace the `Desired capacity` value with your own value in the rest of the walkthrough.

Your command should look like the following example:

```
as-describe-auto-scaling-groups asg-test-termination-policy asg-test-termin
ation-policy  --headers
```

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

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

```
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG           AVAILABILITY-ZONES
  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  asg-test-termination-policy  lc-test-termination-policy
   us-east-1e   1     3     2            OldestInstance
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG

INSTANCE  i-bd9e84c6  us-east-1e  InService   Healthy  lc-test-termination-
policy
INSTANCE  i-bf9e84c4  us-east-1e  InService   Healthy  lc-test-termination-
policy
```

You can see that Auto Scaling launched 2 instances using the `lc-test-termination-policy` launch configuration, and they are Healthy and running (InService). Your Auto Scaling group is associated with your termination policy.

3. **Test if instances are being terminated as specified in your termination policy.**

   To test if your instances are getting terminated as specified in your policy, you'll have to first invoke your Auto Scaling group to scale out (launch a new instance) and then use the termination policy value, `OldestInstance`, specified in the previous step, to scale in (terminate an instance).

   a. **Increase the `Desired capacity` of your Auto Scaling group to invoke scaling out activity.**

      Specify these values for your command:

      - Auto Scaling group name = `asg-test-termination-policy`
      - Desired capacity = `3`

      Your command should look like the following example:

      ```
      as-update-auto-scaling-group asg-test-termination-policy –-desired-capa
      city 3
      ```

      You should get confirmation like the following example:

      ```
      OK-Updated AutoScalingGroup
      ```

   b. **Verify that your Auto Scaling group scaled out to the desired capacity.**

      Your command should look like the following example.

      ```
      as-describe-auto-scaling-groups asg-test-termination-policy --headers
      ```

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

      ```
      AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG            AVAILABILITY-
      ZONES  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
      AUTO-SCALING-GROUP  asg-test-termination-policy  lc-test-termination-
      policy   us-east-1e   1     3     3              NewestInstance
      INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-
      CONFIG
      INSTANCE  i-bd9e84c6   us-east-1e  InService   Healthy  lc-test-termin
      ```

```
ation-policy
INSTANCE  i- bf9e84c4  us-east-1e  InService    Healthy  lc-test-termin
ation-policy
INSTANCE  i-47b5af3c  us-east-1e  InService    Healthy  lc-test-termin
ation-policy
```

Your Auto Scaling group has launched one new instance, `i-47b5af3c`. Make a note of the instance-id of the newly launched instance. You'll use it to verify the scaling activity in the next step.

c. **Update your Auto Scaling group to invoke scaling in activity as specified in your termination policy.**

You can invoke scaling in of your Auto Scaling group by reducing the value of desired capacity.

Specify these values for your command:

- Auto Scaling group name = `asg-test-termination-policy`
- Desired capacity = `2`

Your command should like the following example:

```
as-update-auto-scaling-group asg-test-termination-policy –-desired-capa
city 2
```

You should get confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

d. **Verify that your Auto Scaling group scaled in to the desired capacity.**

In this step you'll verify that a scaling in activity was executed on your Auto Scaling group using the termination policy you specified.

Your command should look like the following example:

```
as-describe-auto-scaling-groups asg-test-termination-policy --headers
```

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

```
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG            AVAILABILITY-
ZONES  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  asg-test-termination-policy  lc-test-termination-
policy   us-east-1e   1     3     2
OldestInstance
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-
CONFIG
INSTANCE  i- bf9e84c4  us-east-1e  Terminating  Healthy lc-test-termina
tion-policy
INSTANCE  i-bd9e84c6   us-east-1e  InService    Healthy lc-test-termina
tion-policy
INSTANCE  i-47b5af3c  us-east-1e  InService    Healthy  lc-test-termina
```

```
tion-policy
```

Your Auto Scaling group description shows that instance `i- bf9e84c4` is terminating. This instance was launched along with `i- bd9e84c6` when Auto Scaling group was created in Step 1. When Auto Scaling executed the termination policy value, `OldestInstance`, for this Auto Scaling group to identify an instance to terminate, it found more than one instance meeting the criteria. Since there were no other policies mentioned, the Default policy was executed.

As per the Default Termination Policy, both the instances were launched using the same launch configuration and were closest to the next instance hour. Auto Scaling then selected a random instance, `i- bf9e84c4`, for termination.

e. **Update your Auto Scaling group with a new termination policy and invoke termination of an instance as specified in your new termination policy.**

You can invoke termination of an instance in your Auto Scaling group by reducing the value of desired capacity.

To decrease the desired capacity and change the termination policy for your Auto Scaling group, use the `as-update-auto-scaling-policy` command by specifying the following values:

- Auto Scaling group name = `asg-test-termination-policy`
- Desired capacity = `1`
- Termination Policy = `NewestInstance`

Your command should like the following example:

```
as-update-auto-scaling-group asg-test-termination-policy –-desired-capa
city 1  --termination-policies "NewestInstance"
```

You should get confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

f. **Verify that your Auto Scaling group scaled in to the desired capacity using the new termination policy.**

In this step you'll verify that a scaling in activity was implemented on your Auto Scaling group using the new termination policy.

Your command should look like the following example:

```
as-describe-auto-scaling-groups asg-test-termination-policy --headers
```

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

```
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG            AVAILABILITY-
ZONES  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  asg-test-termination-policy  lc-test-termination-
policy   us-east-1e   1     3     1
NewestInstance
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-
CONFIG
```

```
INSTANCE  i-bd9e84c6   us-east-1e  InService    Healthy
lc-test-termination-policy
INSTANCE  i-47b5af3c  us-east-1e  Terminating   Healthy  lc-test-termin
ation-policy
```

Your Auto Scaling group description shows that instance `i-47b5af3c` is terminating. Note that the terminating instance was launched when you updated your Auto Scaling group in Step 3e. This is your newest instance.

4. **Clean up.**

   If you were working on your existing Auto Scaling group and want to keep it, you can skip this step. Instead, use `as-update-auto-scaling groups` to set the parameters to the values you want to keep for this Auto Scaling group.

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

   Run the command using the following values:

   - Auto Scaling group name = `asg-test-termination-policy`
   - Force delete (optional parameter) = `--force-delete`

   Your command should look like the following example:

```
as-delete-auto-scaling-group as-test-termination-policy --force-delete
```

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

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

Congratulations! You've successfully created two new termination policies and then you invoked scaling in and scaling out activities to test your new termination policies.

Using the Query API

If you are not familiar with using the Auto Scaling Query API to make Query requests, see Use Query Requests to Call Auto Scaling APIs (p. 20) section in the *Auto Scaling Developer Guide*.

You will use the following Auto Scaling Query API actions to configure your termination policy for your Auto Scaling group.

Actions	Description
CreateLaunchConfiguration	Creates a new launch configuration with specified attributes.

Actions	Description
CreateAutoScalingGroup	Creates a new Auto Scaling group with a specified name and other attributes.
DescribeAutoScalingGroups	Describes the Auto Scaling groups, if the groups exist.
UpdateAutoScalingGroup	Updates the configuration for the specified AutoScalingGroup.
DeleteAutoScalingGroup	Deletes the specified Auto Scaling group, if the group has no instances and no scaling activities are in progress.

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

For detailed descriptions of the Auto Scaling API actions, go to Auto Scaling API Reference.

1. **Create or update your Auto Scaling Group.**

   If you are updating your existing Auto Scaling group to use your Termination Policy, go to step 1c.

   a. **Create launch configuration.**

      If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Auto Scaling Configuration (p. 32). Use the basic scenario to get started using the infrastructure that is typically needed in Auto Scaling.

      To create a new launch configuration to use with your termination policy, call the `CreateLaunchConfiguration` action with the following parameters:

      - LaunchConfigurationName = `lc-test-termination-policy`
      - ImageId = `ami-0078da69`

        If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
      - InstanceType = `m1.small`

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

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

   b. **Create Auto Scaling group.**

      Create your Auto Scaling group by using `CreateAutoScalingGroup` action with the following parameters:

      - AutoScalingGroup name = `asg-test-termination-policy`
      - LaunchConfigurationName = `lc-test-termination-policy`
      - AvailabilityZones = `us-east-1e`
      - MaxSize = 3
      - DesiredCapacity = 2

- MinSize = `1`
- TerminationPolicy = `OldestInstance`

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

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

Skip the following step and go to step 2 to verify that your Auto Scaling group has launched instances with the termination policy.

c. **Update Auto Scaling group.**

Update your existing Auto Scaling group by using the `UpdateAutoScalingGroups` action with the following parameters:

- AutoScalingGroupName = *`your Auto Scaling group name`*
- TerminationPolicy = `OldestInstance`

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>126a833c-f06c-11e1-baa6-f5fc8EXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

2. **Verify that the Auto Scaling group has launched with your termination policy.**

To confirm that Auto Scaling has launched your EC2 instances with your termination policies, call `DescribeAutoScalingGroups` action with the following parameter:

- AutoScalingGroupName = `asg-test-termination-policy`

Note
If you've updated your Auto Scaling group, replace the Auto Scaling group name, `asg-test-termination-policy`, with the name of your Auto Scaling group and replace the `DesiredCapacity` value with your own value, in the rest of the walkthrough.

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
```

```
        <SuspendedProcesses/>
       <AutoScalingGroupName>asg-test-termination-policy</AutoScalingGroup
Name>

        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2012-08-27T20:12:40.335Z</CreatedTime>
        <EnabledMetrics/>
       <LaunchConfigurationName>lc-test-termination-policy</LaunchConfigur
ationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bd9e84c6</InstanceId>
           <LaunchConfigurationName>lc-test-termination-policy</LaunchCon
figurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId> i-bf9e84c4</InstanceId>
           <LaunchConfigurationName>lc-test-termination-policy</LaunchCon
figurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:70eb78a1-687e-4a2b-836c-ef53806746a5:auto
ScalingGroupName/asg-test-term
ination-policy</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>OldestInstance</member>
        </TerminationPolicies>
        <MaxSize>3</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
```

You can see that Auto Scaling launched 2 instances using the `lc-test-termination-policy` launch configuration, and they are Healthy and running (InService). Your Auto Scaling group is associated with your termination policy.

3. **Test if instances are being terminated as specified in your termination policy.**

   To test if your instances are getting terminated as specified in your policy, you'll have to first invoke your Auto Scaling group to scale out (launch a new instance) and then use the termination policy value, `OldestInstance`, specified in the previous step to scale in (terminate an instance).

a. **Increase the `Desired capacity` of your Auto Scaling group to invoke scaling out activity.**

Call `UpdateAutoScalingGroup` action with the following parameters:

- AutoScalingGroupName = `asg-test-termination-policy`
- DesiredCapacity = 3

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>126a833c-f06c-11e1-baa6-f5fc8fEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

b. **Verify that your Auto Scaling group scaled out to the desired capacity.**

To verify that your Auto Scaling group has scaled out as per the updated desired capacity, call `DescribeAutoScalingGroups` by specifying the following parameters:

- AutoScalingGroupName = `asg-test-termination-policy`

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>asg-test-termination-policy</AutoScal
ingGroupName>

        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2012-08-27T17:01:09.850Z</CreatedTime>
        <EnabledMetrics/>
       <LaunchConfigurationName>lc-termination-policy</LaunchConfigura
tionName>

        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bf9e84c4</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bd9e84c6</InstanceId>
          <LaunchConfigurationName>lc-termination-policy</LaunchConfig
```

```
urationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-47b5af3c</InstanceId>
          <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>InService </LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>3</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:9cad1ceb-09ba-468f-aefa-
399aa74a3864:autoScalingGroupName/asg-test-term
ination-policy</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>OldestInstance</member>
        </TerminationPolicies>
        <MaxSize>3</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>2d4116b4-f06c-11e1-82d0-431d8EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Your Auto Scaling group has launched one new instance, `i-47b5af3c`. Make a note of the instance-id of the newly launched instance. You'll use it to verify the scaling activity in the next step.

c. **Update your Auto Scaling group to invoke scaling in activity as specified in your termination policy.**

You can invoke scaling in of your Auto Scaling group by reducing the value of desired capacity.

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

- AutoScalingGroupName = `asg-test-termination-policy`
- DesiredCapacity = 2

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
```

```
    <RequestId>126a833c-f06c-11e1-baa6-f5fc8fEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

d. **Verify that your Auto Scaling group scaled in to the desired capacity.**

In this step you'll verify that a scaling in activity was executed on your Auto Scaling group using the termination policy you specified.

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

- AutoScalingGroupName = `asg-test-termination-policy`

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>asg-test-termination-policy</AutoScal
ingGroupName>

        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2012-08-27T17:01:09.850Z</CreatedTime>
        <EnabledMetrics/>
       <LaunchConfigurationName>lc-termination-policy</LaunchConfigura
tionName>

        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bf9e84c4</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>Terminating</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bd9e84c6</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-47b5af3c</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>InService</LifecycleState>
          </member>
```

```
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:9cad1ceb-09ba-468f-aefa-
399aa74a3864:autoScalingGroupName/asg-test-termination-policy</AutoScal
ingGroupARN>
        <TerminationPolicies>
          <member>OldestInstance</member>
        </TerminationPolicies>
        <MaxSize>3</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>c4ea02b1-f06c-11e1-a139-71b5eEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Your Auto Scaling group description shows that instance `i-bf9e84c4` is terminating. This instance was launched along with `i-bd9e84c6` when Auto Scaling group was created in Step 1. When Auto Scaling executed the termination policy value, `OldestInstance`, for this Auto Scaling group to identify an instance to terminate, it found more than one instance meeting the criteria. Since there were no other policies mentioned, the Default policy was executed.

As per the Default Termination Policy, both the instances were launched using the same launch configuration and were closest to the next instance hour. Auto Scaling then selected a random instance, `i-bf9e84c4`, for termination.

e. **Update your Auto Scaling group with a new termination policy and invoke termination of an instance as specified in your new termination policy.**

You can invoke termination of an instance in your Auto Scaling group by reducing the value of desired capacity.

To decrease the desired capacity and change the termination policy for your Auto scaling group, call the `UpdateAutoScalingGroups` command by specifying the following values:

- AutoScalingGroupName = `asg-test-termination-policy`
- DesiredCapacity = `1`
- TerminationPolicy = `NewestInstance`

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>126a833c-f06c-11e1-baa6-f5fc8fEXAMPLE</RequestId>
```

```
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

f. **Verify that your Auto Scaling group scaled in to the desired capacity using the new termination policy.**

In this step, you'll verify that a scaling in activity was executed on your Auto Scaling group using the new termination policies by calling the `DescribeAutoScalingGroups` action and specifying the following parameter:

- AutoScalingGroupName = `asg-test-termination-policy`

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>asg-test-termination-policy</AutoScal
ingGroupName>

        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2012-08-27T17:01:09.850Z</CreatedTime>
        <EnabledMetrics/>
       <LaunchConfigurationName>lc-termination-policy</LaunchConfigura
tionName>

        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-bd9e84c6</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-47b5af3c</InstanceId>
           <LaunchConfigurationName>lc-termination-policy</LaunchConfig
urationName>
            <LifecycleState>Terminating</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
```

```
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScal
ingGroup:9cad1ceb-09ba-468f-aefa-399aa74a3864:autoScalingGroupName/asg-
test-termination-policy</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>NewestInstance</member>
        </TerminationPolicies>
        <MaxSize>3</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>fd525c2b-f06d-11e1-b584-cddb6EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

Your Auto Scaling group description shows that instance `i-47b5af3c` is terminating. Note that the terminating instance was launched when you updated your Auto Scaling group in Step 3e. This is your newest instance.

4. **Clean up.**

   If you were working on your existing Auto Scaling group and want to keep it, you can skip this step. Instead, call the `UpdateAutoScalingGroups` to set the parameters to the values you want to keep for this Auto Scaling group.

   After you're finished using your instances and your Auto Scaling group for testing your termination policy, it is a good practice to clean up. Call the `DeleteAutoScalingGroup` action by setting the optional `ForceDelete` parameter to `true`. `ForceDelete` specifies that EC2 instances that are part of the Auto Scaling group will be terminated with the Auto Scaling group, even if the instances are still running. If you don't set the `ForceDelete` parameter to `true`, then you cannot delete your Auto Scaling group until you have terminated all instances running in that Auto Scaling group.

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

   - AutoScalingGroupName = `asg-test-termination-policy`
   - ForceDelete (optional parameter) = `true`

   If your request is successful, you should see a prompt like in the following example:

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

   Type y to confirm that you want to delete the Auto Scaling group. After you confirm, Auto Scaling deletes the group. You should get a confirmation like the following example:

```
<DeleteAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>c7f33164-f071-11e1-a5a8-e129b4EXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteAutoScalingGroupResponse>
```

Congratulations! You've successfully created two new termination policies and then you invoked scaling in and scaling out activities to test your new termination policies.

Tag Your Auto Scaling Groups and Amazon EC2 Instances

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

Additional Features

Optionally, you can propagate Auto Scaling group tags to the Amazon EC2 instances launched by Auto Scaling. The Auto Scaling group tags that you propagate to Amazon EC2 instances can be used like any other Amazon EC2 instance tag, including cost allocation. For more information about cost allocation, see Cost Allocation and Tagging in the *AWS Account Billing*.

When you use the `PropagateAtLaunch` flag with the `as-create-or-update-tags` command, the tag you create will be applied to new Amazon EC2 instances launched by the Auto Scaling group. Likewise, when you modify a tag, the updated version will be applied to new instances launched by the Auto Scaling group after the change. Tags created or modified with the `as-create-or-update-tags` command will not apply to instances that are already running at the time you used the command. For an example, go to Change the Tag (p. 97).

> **Note**
> When you launch instances in an Auto Scaling group, Auto Scaling automatically tags each one with the group name. This tag can be identified by its key, `aws:autoscaling:groupName`. Tags containing the prefix `aws:` have been created by AWS, cannot be edited or deleted, and do not count against your limit of 10 tags per instance.

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

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

Tools You Will Use

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

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

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

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

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Auto Scaling Configuration (p. 32). Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

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

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

Your command should look similar to the following example:

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

You should get a confirmation like the following example:

```
OK-Created launch config
```

Create a Tag for Your Auto Scaling Group

Create your Auto Scaling group with a tag and specify that the tag also applies to the instances launched by the Auto Scaling group. You use the `--tag` option to define the tag for your Auto Scaling group and instances. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 34).

Specify these values for the command:

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

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

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

Provide these values for the following tag options:

Option	Description	Example value
`k`	Tag's key, as part of key-value pair. Always required.	`version`
`v`	Tag's value, as part of key-value pair. Optional.	`1.0`
`t`	The type of resource to which the tag is applied. Not required with `as-create-auto-scaling-group` because the tag is being created with the Auto Scaling group resource.	Not specified with the `as-create-auto-scaling-group` command.
`id`	The name of the resource to which the tag is applied. Not required with `as-create-auto-scaling-group` because the tag is being created with the Auto Scaling group resource.	Not specified with the `as-create-auto-scaling-group` command.
`p`	The propagate-at-launch flag. Specify this flag only if you want the tags to be applied to newly launched Amazon EC2.	`true`

Your command should look like the following example:

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

You should get confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Confirm Tag Creation

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

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

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

1. Use the `as-describe-tags` command to verify that the tag is created. The command takes the following arguments:

   as-describe-tags [--filter "key1=*value1*,key2=*value2*..." [--filter "key1=*value1*,key2=*value2*..." ...]] [General Options]

   The `key1`, `key2` filter options are the resource name, resource type, tag key, tag value, and the propagate flag that you used to define your tag. For this walkthrough, specify the following:

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

   Your command should look like the following example:

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

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

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

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

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

   Your command should look like the following example:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Change the Tag

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

The `as-create-or-update-tags` command takes the following arguments:

`as-create-or-update-tags --tag "id=value,t=value, k=value, [v=value], [p=value]" [--tag "id=value, t=value, k=value, [v=value], [p=value]]" ...] [General Options]`

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

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

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

You should receive a confirmation similar to the following example:

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

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

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

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

Your command should look similar to the following example:

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

The confirmation should look like the following example:

```
OK-Updated AutoScalingGroup
```

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

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

Your results should be similar to the following example:

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

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

```
RESERVATION r-45253524 629715795501 default
INSTANCE i-33698556 ami-31814f58 ec2-23-20-37-36.compute-1.amazonaws.com ip-10-
32-153-166.ec2.internal running  0  m1.small 2012-01-23T21:41:44+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 23.20.37.36 10.32.153.166   ebs     para
virtual xen 29849241-1242-4524-ba07-09248cd29652 sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-2f303942 2012-01-23T21:42:12.000Z
TAG instance i-33698556 version 1.0
TAG instance i-33698556 aws:autoscaling:groupName MyTagASG
RESERVATION r-b1ebf9d0 629715795501 default
INSTANCE i-5346af36 ami-31814f58 ec2-107-22-159-5.compute-1.amazonaws.com ip-
10-204-54-176.ec2.internal running  0  m1.small 2012-01-24T07:17:50+0000 us-
east-1a aki-805ea7e9   monitoring-enabled 107.22.159.5 10.204.54.176   ebs
 paravirtual xen a848fb19-703f-4579-9ee4-930b2f1e5a9f sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-b3aea6de 2012-01-24T07:18:10.000Z
TAG instance i-5346af36 aws:autoscaling:groupName MyTagASG
TAG instance i-5346af36 version 2.0
RESERVATION r-b7ebf9d6 629715795501 default
INSTANCE i-5946af3c ami-31814f58 ec2-50-16-11-77.compute-1.amazonaws.com ip-10-
118-54-85.ec2.internal running  0  m1.small 2012-01-24T07:17:52+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 50.16.11.77 10.118.54.85   ebs    paravir
tual xen bfa8068e-a3a9-4374-8db0-7550010f477f sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-8daea6e0 2012-01-24T07:18:15.000Z
TAG instance i-5946af3c version 2.0
TAG instance i-5946af3c aws:autoscaling:groupName MyTagASG
RESERVATION r-bbebf9da 629715795501 default
INSTANCE i-2546af40 ami-31814f58 ec2-50-19-152-11.compute-1.amazonaws.com ip-
10-116-241-173.ec2.internal running  0  m1.small 2012-01-24T07:17:53+0000 us-
east-1a aki-805ea7e9   monitoring-enabled 50.19.152.11 10.116.241.173   ebs
  paravirtual xen 7d419c3a-0527-4970-80f5-b6467ce44849 sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-89aea6e4 2012-01-24T07:18:18.000Z
TAG instance i-2546af40 version 2.0
TAG instance i-2546af40 aws:autoscaling:groupName MyTagASG
RESERVATION r-bfebf9de 629715795501 default
INSTANCE i-2346af46 ami-31814f58 ec2-184-72-167-153.compute-1.amazonaws.com ip-
10-203-54-74.ec2.internal running  0  m1.small 2012-01-24T07:17:54+0000 us-east-
1a aki-805ea7e9   monitoring-enabled 184.72.167.153 10.203.54.74   ebs
paravirtual xen ec0221df-0274-44e1-aa13-04b7f17442ef sg-286ee441 default
BLOCKDEVICE /dev/sda1 vol-8baea6e6 2012-01-24T07:18:15.000Z
TAG instance i-2346af46 aws:autoscaling:groupName MyTagASG
TAG instance i-2346af46 version 2.0
```

Clean-up

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

Run the command with the following values:

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

Your command should look like the following example:

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

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

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

Tasks Completed

You just performed the following tasks:

- Created a launch configuration.
- Created an Auto Scaling group with a tag.
- Confirmed that the Auto Scaling group has a tag and that the tag is applied to instances that the Auto Scaling group launched.
- Changed the tag and Auto Scaling group, and confirmed that the changes are propagated to new instances launched.
- Cleaned up.

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

> **Note**
> The instance associated with the Auto Scaling group you just created is not launched immediately. So, if you run the snippet as a single code block, it could take a few minutes before you see the instance information.

```
as-create-launch-config MyTagLC --image-id ami-31814f58 --instance-type m1.small
as-create-auto-scaling-group MyTagASG --launch-configuration MyTagLC --availab
ility-zones us-east-1a --min-size 1 --max-size 10 --tag "k=version, v=1.0,
p=true"
as-describe-tags --filter "key1=version, key2=1.0"
as-describe-auto-scaling-groups MyTagASG
as-describe-auto-scaling-instances
ec2-describe-instances i-33698556
as-create-or-update-tags --tag "id=MyTagASG, t=auto-scaling-group, k=version,
v=2.0, p=true"
as-update-auto-scaling-group MyTagASG --min-size 5
as-describe-auto-scaling-instances
ec2-describe-instances
as-delete-auto-scaling-group MyTagASG --force-delete
```

Launch Auto Scaling Instances into Amazon VPC

You can create an Auto Scaling group that launches instances into the Amazon Virtual Private Cloud (Amazon VPC).

Amazon VPC is a virtual network you can design that resembles a network running out of a data center. Amazon VPC enables you to create an isolated portion of the Amazon Web Services cloud where you can launch Amazon EC2 instances that have private addresses in the range of your choice. You can create a VPC that spans multiple Availability Zones. You can define subnets within your VPC that you can use to group similar kinds of instances based on IP address range. Each subnet must reside entirely within one Availability Zone. It is possible for you to define multiple subnets within an Availability zone, but Auto Scaling requires you to define one subnet per Availability Zone per Auto Scaling group.

For more information on Amazon VPC, see What is Amazon VPC in the *Amazon Virtual Private Cloud User Guide*. For information on Amazon VPC subnets, see Your VPC and Subnets.

In this section, we will walk you through the following steps for launching your basic Auto Scaling infrastucture within EC2-VPC platform:

1. Create an Amazon VPC.
2. Create subnets in your Amazon VPC.
3. Create a launch configuration.
4. Create Auto Scaling groups in your Amazon VPC subnets.
5. Check that instances are launched in the subnets we specified.
6. Delete your Auto Scaling group.

Tools You Will Use

In this walkthrough exercise, when you create an Amazon VPC and subnets, you'll use Amazon EC2 command line tool commands, and we'll point you to documentation on how to use them. For the Auto Scaling tasks, you'll use the Auto Scaling CLI commands, which we'll discuss in more detail.

Amazon EC2 CLI Commands	Description
`ec2-create-vpc`	Creates a VPC with a CIDR block you specify.
`ec2-create-subnet`	Creates a subnet in an existing VPC.
`ec2-describe-subnets`	Gives you information about your subnets.
`ec2-describe-instances`	Returns information about instances that you own.

Auto Scaling CLI Commands	Description
`as-create-launch-config`	Creates a new launch configuration with specified attributes.
`as-create-auto-scaling-group`	Creates a new Auto Scaling group with specified name and other attributes.
`as-describe-auto-scaling-groups`	Describes the specified Auto Scaling group(s) if the group exists.
`as-describe-scaling-activities`	Describes a set of activities or all activities belonging to an Auto Scaling group.
`as-delete-auto-scaling-group`	Delete the specified auto scaling group if the group has no instances and no scaling activities in progress.

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

Create Your Amazon Virtual Private Cloud (Amazon VPC)

First, you need to create your Amazon Virtual Private Cloud (Amazon VPC) using the `ec2-create-vpc` command. For information about `ec2-create-vpc`, see ec2-create-vpc in the *Amazon Elastic Compute Cloud CLI Reference*. For information about Amazon VPC, see Introduction to the Amazon Virtual Private Cloud in the *Amazon Virtual Private Cloud User Guide*.

In the following example you use the `ec2-create-vpc` command to create a VPC.

```
ec2-create-vpc "10.0.0.0/24" --region ap-southeast-1
```

The response includes your VPC ID.

```
VPC vpc-2e00c747 available 10.0.0.0/24 dopt-2200c74b default
```

Create Subnets in Your Amazon VPC

You cannot create your subnet until you have created your Amazon VPC. Since you've just created your Amazon VPC, you're ready to use the `ec2-create-subnet` command to create a subnet; specify the VPC ID from the previous step.

```
ec2-create-subnet --vpc vpc-2e00c747 -cidr "10.0.0.0/28" --availability-zone
ap-southeast-1a --region ap-southeast-1
```

For information about `ec2-create-subnet`, see ec2-create-subnet in the *Amazon Elastic Compute Cloud CLI Reference*.

The subnet will be created in your VPC in the Availability Zone you specified.

```
SUBNET subnet-530fc83a available vpc-2e00c747 10.0.0.0/28 11 ap-southeast-1a
ap-southeast-1
```

Create a subnet in a different Availability Zone.

```
ec2-create-subnet --vpc vpc-2e00c747 -cidr "10.0.0.128/28" --availability-zone
 ap-southeast-1b --region ap-southeast-1
```

The subnet will be created in your VPC in the Availability Zone you specified.

```
SUBNET subnet-610acd08 available vpc-2e00c747  10.0.0.128/28 11 ap-southeast-
1a
```

Get a list of subnets in your VPC.

```
ec2-describe-subnets --region ap-southeast-1
```

You will get information about the two subnets you just created in your VPC.

```
SUBNET subnet-610acd08 available vpc-2e00c747  10.0.0.128/28 11 ap-southeast-
1b ap-southeast-1
SUBNET subnet-530fc83a available vpc-2e00c747 10.0.0.0/28 11 ap-southeast-1a
ap-southeast-1
```

Create Launch Configuration

After you've created your Amazon VPC and subnets, you define the Auto Scaling scenario. Create a launch configuration, or use an existing launch configuration.

The Auto Scaling launch configuration specifies the template that Auto Scaling uses to launch Amazon EC2 instances. This template contains all the information necessary for Auto Scaling to launch instances that run your application. If you've never created a launch configuration before, go to Create a Launch Configuration (p. 33).

Specify the following values for the required options:

- Launch configuration name: `myvpclc`
- Instance type: `m1.small`
- Image ID: `ami-b4b0cae6`

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

In addition, specify the `--region` general option:

- Region: `ap-southeast-1`

  Note
  The AMI or image ID you specify must be supported in the region of your Amazon VPC.

Your `as-create-launch-config` command will look like this:

```
as-create-launch-config myvpclc --image-id ami-b4b0cae6 --instance-type m1.small
 --region ap-southeast-1
```

You will get a confirmation that your launch configuration was created successfully.

```
OK-Created launch config
```

Create Auto Scaling Group

An Auto Scaling group is a collection of Amazon EC2 instances. You can specify settings like the minimum, maximum, and desired number of EC2 instances for an Auto Scaling group to which you want to apply certain scaling actions. For information on how to create an Auto Scaling group, go to Create an Auto Scaling Group (p. 34).

Specify these values for the following options:

- Auto Scaling group name: `myvpcasgroup`

- Launch configuration name: `myvpclc`
- Availability Zone: `"ap-southeast-1a,ap-southeast-1b"`
- Minimum size: `0`
- Maximum size: `10`
- Desired capacity: `10`

To launch instances in the subnets you created in your Amazon VPC, you also must specify the following zone identifier and region:

- VPC zone identifier: `"subnet-610acd08,subnet-530fc83a"`

  Important
  Do NOT use the VPC identifier to specify the VPCZoneIdentifier parameter. Amazon VPC has both a VPC identifier (e.g., vpc-1a2b3c4d) and a subnet identifier (e.g., subnet-9d4a7b6c). You must use the subnet identifier instead of the VPC identifier.

  Note
  The Availability Zone of the subnet must match the Availability Zone you specify. In addition, when you specify several subnets and Availability Zones separated by commas (,) make sure that there is no space between the comma and the next item.

- Region: `ap-southeast-1`

Your request will look like this:

```
as-create-auto-scaling-group myvpcasgroup --launch-configuration myvpclc --
availability-zones
"ap-southeast-1b,ap-southeast-1a" --min-size 0 --max-size 10 --desired-capacity
 10 --vpc-zone-identifier
"subnet-610acd08,subnet-530fc83a" --region ap-southeast-1
```

The response should look something like this.

```
OK-Created AutoScalingGroup
```

Confirm that Instances Launched in Subnets

To verify that your Auto Scaling group successfully launched instances in the Availability Zone you specified in your Amazon VPC subnets, you can use the following commands:

- `as-describe-auto-scaling-groups`
- `as-describe-scaling-activities`
- `ec2-describe-instances`

The `as-describe-auto-scaling-groups` command describes the Auto Scaling group, if it exists. Run this command to confirm that the Auto Scaling group `myvpcasgroup` was created and is launching instances according to your specifications.

```
as-describe-auto-scaling-groups myvpcasgroup --region ap-southeast-1 -H
```

Our example response confirms that the Auto Scaling group `myvpcasgroup` was created and that instances were launched according to specifications.

```
AUTO-SCALING-GROUP  GROUP-NAME    LAUNCH-CONFIG  AVAILABILITY-ZONES
    MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  myvpcasgroup  myvpclc        ap-southeast-1b,ap-southeast-
1a  0         10        10
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
INSTANCE  i-e5f4c3b0   ap-southeast-1b    InService  Healthy  myvpclc
INSTANCE  i-e7f4c3b2   ap-southeast-1b    InService  Healthy  myvpclc
INSTANCE  i-eff4c3ba   ap-southeast-1b    InService  Healthy  myvpclc
INSTANCE  i-e9f4c3bc   ap-southeast-1b    InService  Healthy  myvpclc
INSTANCE  i-ebf4c3be   ap-southeast-1b    InService  Healthy  myvpclc
INSTANCE  i-95f4c3c0   ap-southeast-1a    InService  Healthy  myvpclc
INSTANCE  i-97f4c3c2   ap-southeast-1a    InService  Healthy  myvpclc
INSTANCE  i-9df4c3c8   ap-southeast-1a    InService  Healthy  myvpclc
INSTANCE  i-93f4c3c6   ap-southeast-1a    InService  Healthy  myvpclc
INSTANCE  i-91f4c3c4   ap-southeast-1a    InService  Healthy  myvpclc
```

The `as-describe-scaling-activities` command lists the scaling activities for the Auto Scaling group. When you run this command, it lists the activities that took place for the `myvpcasgroup`.

```
as-describe-scaling-activities --auto-scaling-group myvpcasgroup --region ap-
southeast-1 -H
```

The response shows 10 activities that were successfully run within the timespan of a minute for Auto Scaling group `myvpcasgroup`.

```
ACTIVITY  ACTIVITY-ID                            END-TIME             GROUP-
NAME    CODE
ACTIVITY  188b5c5f-79fc-4dcc-95a3-2762eba64702  2011-12-12T22:46:08Z  myvpcas
group  Successful
ACTIVITY  bc35c15e-cfb1-491c-b4e5-4bc99bf1f68c  2011-12-12T22:46:24Z  myvpcas
group  Successful
ACTIVITY  b30546fb-b4a7-47fb-bb20-3e8f977bfbdd  2011-12-12T22:46:20Z  myvpcas
group  Successful
ACTIVITY  18be6da1-7448-4548-97f7-fe1d6a86538a  2011-12-12T22:46:19Z  myvpcas
group  Successful
ACTIVITY  7952cf76-b3d2-4faf-b725-137d79bb5769  2011-12-12T22:46:21Z  myvpcas
group  Successful
ACTIVITY  ad501827-5b10-4479-aadf-6b6ef6eb22bc  2011-12-12T22:46:19Z  myvpcas
group  Successful
ACTIVITY  90cad1e5-c64a-44b9-a619-25440fa411ad  2011-12-12T22:46:08Z  myvpcas
group  Successful
ACTIVITY  3acc0e73-17fa-4326-9a16-11ec65295456  2011-12-12T22:46:38Z  myvpcas
group  Successful
ACTIVITY  6865beb5-98da-4377-a684-b727be1cc686  2011-12-12T22:46:38Z  myvpcas
group  Successful
ACTIVITY  cf072f5a-021b-4c1a-8557-827e19af3a76  2011-12-12T22:46:19Z  myvpcas
group  Successful
```

The `ec2-describe-instances` command describes instances that were launched. In this walkthrough you run the command, specifying the instance ID of one of the 10 instances launched by the `myvpcasgroup`, with the verbose flag general option (`-v` or `--verbose`) to see if the private IP address of the instance is in the subnet `subnet-610acd08`.

```
ec2-describe-instances i-e5f4c3b0 --region ap-southeast-1 --verbose
```

Here is part of the response.

```
RESERVATION r-ad623af8 629715795501
INSTANCE i-e5f4c3b0 ami-b4b0cae6   running  0  m1.small 2011-12-12T22:45:32+0000
 ap-southeast-1b aki-a4225af6   monitoring-enabled  10.0.0.136 vpc-2e00c747
subnet-610acd08 ebs     paravirtual xen cf072f5a-021b-4c1a-8557-827e19af3a76
sg-a19f8ccd default
BLOCKDEVICE /dev/sda1 vol-2eeea540 2011-12-12T22:46:08.000Z
TAG instance i-e5f4c3b0 aws:autoscaling:groupName myvpcasgroup
REQUEST ID 30e8d688-652f-4a1b-b772-ede8e4a4e72c
```

Delete the Auto Scaling Group

After you're done using your instances and Auto Scaling group, it is good practice to clean up. An efficient way to perform this task is by running the `as-delete-auto-scaling-group` command with the optional `--force-delete` parameter. *Force delete* specifies that instances of the Auto Scaling group will be deleted together with the Auto Scaling group even if the instances are still running. If you don't specify `--force-delete` parameter when you cannot delete your Auto Scaling group until you've manually stopped and terminated all instances of that Auto Scaling group.

Run the command with the following values:

- Auto Scaling group name: `myvpcasgroup`
- Region (general option): `--region ap-southeast-1`
- Force delete (optional parameter): `--force-delete`

This is how your command will look.

```
as-delete-auto-scaling-group myvpcasgroup --force-delete --region ap-southeast-
1
```

Confirm that you want to delete the Auto Scaling group and it will be deleted.

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

Tasks Completed

You just performed the following tasks:

- Created an Amazon VPC.
- Created two subnets in the Amazon VPC.
- Created a launch configuration.
- Created an Auto Scaling group.
- Confirmed that your Auto Scaling group exists.
- Checked that your Auto Scaling group contains running instances in the subnet you created.
- Deleted your Auto Scaling group.

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

```
ec2-create-vpc "10.0.0.0/24" --region ap-southeast-1
ec2-create-subnet --vpc vpc-2e00c747 -cidr "10.0.0.0/28" --availability-zone
ap-southeast-1a --region ap-southeast-1
ec2-create-subnet --vpc vpc-2e00c747 -cidr "10.0.0.128/28" --availability-zone
 ap-southeast-1b --region ap-southeast-1
as-create-launch-config myvpclc --image-id ami-b4b0cae6 --instance-type m1.small
 --region ap-southeast-1
as-create-auto-scaling-group myvpcasgroup --launch-configuration myvpclc --
availability-zones
"ap-southeast-1b,ap-southeast-1a" --min-size 0 --max-size 10 --desired-capacity
 10 --vpc-zone-identifier
"subnet-610acd08,subnet-530fc83a" --region ap-southeast-1
as-describe-auto-scaling-groups myvpcasgroup --region ap-southeast-1 -H
as-describe-scaling-activities --auto-scaling-group myvpcasgroup --region ap-
southeast-1 -H
ec2-describe-instances i-e5f4c3b0 --region ap-southeast-1 --verbose
as-delete-auto-scaling-group myvpcasgroup --force-delete --region ap-southeast-
1
```

Configure Health Checks for Your Auto Scaling Group

Auto Scaling ensures that the EC2 instances within the Auto Scaling group are running and in good shape by periodically performing health checks on the instances. When Auto Scaling determines that an instance is unhealthy, it terminates that instance and launches a new one. This helps in maintaining the number of running instances at the minimum number (or desired number, if specified) that you defined.

By default, your Auto Scaling group determines the health state of each instance by periodically checking the results of Amazon EC2 instance status checks. If you have associated your Auto Scaling group with an Elastic Load Balancing load balancer and have chosen to use the Elastic Load Balancing health check, Auto Scaling will determine the health status of the instances by checking the results of both the Amazon EC2 instance status checks and the Elastic Load Balancing instance health checks.

If you have your own health check system, you can use the information from your health check system to set the health state of the instances in the Auto Scaling group.

The following sections step you through the process for explicitly specifying the health state of an instance in your Auto Scaling group. You can use either the Auto Scaling command line interface (CLI) or the Query API to complete this task.

Using the Command Line Interface

You can use the Auto Scaling CLI to configure the health state of an instance and then verify the instance's health state.

To configure the health state of an instance

1. Use the `as-set-instance-health` command by specifying the following values:

   - Instance id = *`i-123abc45d`*
   - Health Status = `Unhealthy`

   Your command should look similar to the following example:

```
as-set-instance-health i-123abc45d –-status Unhealthy
```

You should get a confirmation like the following example:

```
OK-Instance Health Set
```

2. Use the `as-describe-auto-scaling-groups` command to verify if the instance is marked `Unhealthy`. You can optionally specify the following value:

   - Auto Scaling group name = *my-test-asg*

   Your command should look similar to the following example:

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

   You should get a response that includes the details of the instances in the Auto Scaling group, as in the following example:

```
........
INSTANCE  i-123abc45d us-east-1a  Terminating  Unhealthy  my-test-lc
........
```

Auto Scaling has marked the instance `Unhealthy` and has started terminating the instance.

Using the Query API

You can use the Auto Scaling Query API to configure the health state of an instance and then verify the instance's health state.

To configure the health state of an instance

1. Call the SetInstanceHealth action by specifying the following parameters:

   - InstanceId = *i-123abc45d*
   - HealthStatus = `Unhealthy`

   Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?InstanceId=i-123abc45d
&ShouldRespectGracePeriod=true
&HealthStatus=Unhealthy
&Version=2011-01-01
&Action=SetInstanceHealth
&AUTHPARAMS
```

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

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

```
  <ResponseMetadata>
    <RequestId>a3603249-ad22-11e2-bada-31b1bEXAMPLE</RequestId>
  </ResponseMetadata>
  </SetInstanceHealthResponse>
```

2. Call the DescribeAutoScalingGroups action to verify if the instance is marked `Unhealthy`. You can optionally specify the following parameter:

   - AutoScalingGroupNames.member.1 = *`my-test-asg`*

   Your request should look similar to the following example:

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

   The response includes details of the instances in the Auto Scaling group, as in the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Unhealthy</HealthStatus>
            <AvailabilityZone>us-east-1c</AvailabilityZone>
            <InstanceId>i-i-123abc45d</InstanceId>
            <LaunchConfigurationName>my-test-lc1</LaunchConfigurationName>
            <LifecycleState>Terminating</LifecycleState>
          </member>
   ............
   ............
```

Auto Scaling has marked the instance `Unhealthy` and has started terminating the instance.

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

To merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones, rezone one of the single-zone groups into a multi-zone Auto Scaling group, and then delete the other Auto Scaling groups.

This section walks you through the process of merging separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones.

The following examples assume that you have two identical groups: `myGroupA` in Availability Zone `us-east-1a`, and `myGroupB` in Availability Zone `us-east-1c`. The two Auto Scaling groups have the following specifications:

- Minimum Size = 2
- Maximum Size = 5
- Desired Capacity = 5

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

You can merge separate single-zone Auto Scaling groups into a single Auto Scaling group spanning multiple Availability Zones using the Auto Scaling command line interface (CLI) or the Query API.

Topics

Using the Command Line Interface

Before you begin, be sure you have installed the CLI tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14).

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

1. Use the `as-update-auto-scaling-group` command and specify the following values:

   - Auto Scaling group name = `myGroupA`
   - Availability Zones = `us-east-1a, us-east-1c`
   - Min size = `4`
   - Max size = `10`

   Your command should look similar to the following example:

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

   You should get a confirmation similar to the following example:

```
OK-AutoScaling Group updated
```

2. Use the `as-set-desired-capacity` command to increase the capacity of `myGroupA` to six. Specify the following values:

   - Auto Scaling group name = `myGroupA`
   - Desired capacity = `6`

   Your command should look similar to the following example:

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

3. Use the `as-describe-auto-scaling-groups` command to check that `myGroupA` has been successfully added to the additional zones and is at the required capacity. Specify the following values:

   - Auto Scaling group name = `myGroupA`

   Your command should look similar to the following example:

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

4. After you've confirmed that your Auto Scaling group `myGroupA` has expanded to an additional Availability Zone `us-east-1c`, you can delete the `myGroupB` Auto Scaling group.

   a. To delete an Auto Scaling group

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

      - Auto Scaling group name = `myGroupB`
      - Min size = `0`
      - Max size = `0`

      Your command should look similar to the following example:

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

      You should get a confirmation similar to the following example:

```
OK-AutoScaling Group updated
```

   b. Use the `as-describe-auto-scaling-groups` command to verify that no instances remain in your Auto Scaling group. Specify the following values:

      - Auto Scaling group name = `myGroupB`

      Your command should look similar to the following example:

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

Auto Scaling returns the following:

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

c. Use the `as-describe-scaling-activities` command to verify that your scaling activities are successful. Specify the following value:

- Auto Scaling group name = `myGroupB`

Your command should look similar to the following example:

```
PROMPT>as-describe-scaling-activities myGroupB
```

Auto Scaling returns the following:

```
ACTIVITY  ACTIVITY-ID           END-TIME             CODE        CAUSE
ACTIVITY  74758a33-bfd5-4df...  2009-05-11T16:27:36Z Successful  "At
2009-05-21 10:00:00Z an instance was shutdown."
ACTIVITY  74958a77-bfd5-4df...  2009-05-11T16:27:36Z Successful  "At
2009-05-21 10:00:00Z an instance was shutdown."
```

d. Use the `as-delete-auto-scaling-group` command and specify the following value:

- Auto Scaling group name = `myGroupB`

Your command should look similar to the following example:

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

Auto Scaling returns the following:

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

- Enter `y` to delete the Auto Scaling group.

Auto Scaling returns the following:

```
OK-Deleted AutoScalingGroup
```

Using the Query API

You can merge your separate single-zone Auto Scaling groups into a single multi-zone group using the Query API. For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

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

1. Call the UpdateAutoScalingGroup action with the following parameters to add an additional Availability Zone to the myGroupA Auto Scaling group:

   - *AvailabilityZones.member.1* = us-east-1a
   - *AvailabilityZones.member.2* = us-east-1c
   - *AutoScalingGroupName* = myGroupA
   - *MinSize* = 4
   - *MaxSize* = 10

2. Call the SetDesiredCapacity action, and specify the following parameters to increase the capacity of myGroupA to six:

   - *AutoScalingGroupName* = myGroupA
   - *DesiredCapacity* = 6

3. Call the DescribeAutoScalingGroups action, and specify the following parameter to verify an additional zone has been successfully added to the myGroupA Auto Scaling group and the group is at the required capacity:

   - *AutoScalingGroupName* = myGroupA

4. After you've confirmed that your Auto Scaling group myGroupA has expanded to an additional Availability Zone us-east-1c, you can delete the myGroupB Auto Scaling group.
   To delete myGroupB:

   a. Call UpdateAutoScalingGroup action, and specify the following parameters:

      - *AutoScalingGroupName* = myGroupB
      - *MinSize* = 0
      - *MaxSize* = 0

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

      i. Call the DescribeAutoScalingGroups action, and specify the following parameter to verify that no instances are left in the group.

         - *AutoScalingGroupName* = myGroupB

      ii. Call the DescribeScalingActivites action, and specify the following parameter to verify that all scaling activities have completed.

         - *AutoScalingGroupName* = myGroupB

   c. Call the DeleteAutoScalingGroup action, and specify the following parameter:

- *AutoScalingGroupName* = myGroupB

Suspend and Resume Auto Scaling Process

Auto Scaling allows you to suspend and then resume one or more of the Auto Scaling processes in your Auto Scaling group. This can be very useful when you want to investigate a configuration problem or other issue with your web application and then make changes to your application, without triggering the Auto Scaling process.

There are two primary Auto Scaling process types — `Launch` and `Terminate`. The `Launch` process creates a new Amazon EC2 instance for an Auto Scaling group, and the `Terminate` process removes an existing Amazon EC2 instance.

The remaining Auto Scaling process types relate to specific Auto Scaling features:

- AddToLoadBalancer
- AlarmNotification
- AZRebalance
- HealthCheck
- ReplaceUnhealthy
- ScheduledActions

If you suspend `Launch` or `Terminate`, all other process types are affected to varying degrees. The following descriptions discuss how each process type is affected by a suspension of a `Launch` or a `Terminate` process.

The `AddToLoadBalancer` process type adds instances to the the load balancer when the instances are launched. If you suspend this process, Auto Scaling will launch the instances but will not add them to the load balancer. If you resume the `AddToLoadBalancer` process, Auto Scaling will also resume adding new instances to the load balancer when they are launched. However, Auto Scaling will not add running instances that were launched while the process was suspended; those instances must be added manually using the RegisterInstancesWithLoadBalancer Elastic Load Balancing API action or the `elb-register-instances-with-lb` Elastic Load Balancing command.

The `AlarmNotification` process type accepts notifications from Amazon CloudWatch alarms that are associated with the Auto Scaling group. If you suspend the `AlarmNotification` process type, Auto Scaling will not automatically execute scaling policies that would be triggered by alarms.

Although the `AlarmNotification` process type is not directly affected by a suspension of `Launch` or `Terminate`, alarm notifications are often used to signal that a change in the size of the Auto Scaling group is warranted. If you suspend `Launch` or `Terminate` process, Auto Scaling might not be able to implement the alarm's associated policy.

The `AZRebalance` process type seeks to maintain a balanced number of instances across Availability Zones within a region. If you remove an Availability Zone from your Auto Scaling group or an Availability Zone otherwise becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone before terminating the unhealthy or unavailable instances. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the designated Availability Zones.

If you suspend the `Launch` process, the `AZRebalance` process will neither launch new instances nor terminate existing instances. This is because the `AZRebalance` process terminates existing instances only after launching the replacement instances.

If you suspend the `Terminate` process, the `AZRebalance` process can cause your Auto Scaling group to grow up to ten percent larger than the maximum size. This is because Auto Scaling allows groups to temporarily grow larger than the maximum size during rebalancing activities. If Auto Scaling cannot terminate instances, your Auto Scaling group could remain up to ten percent larger than the maximum size until you resume the terminate process type.

The `HealthCheck` process type checks the health of the instances. Auto Scaling marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing informs Auto Scaling that the instance is unhealthy. The `HealthCheck` process can override the health status of an instance that you set with SetInstanceHealth API action or the `as-set-instance-health` command.

The `ReplaceUnhealthy` process terminates instances that are marked as unhealthy and subsequently creates new instances to replace them. This process calls both of the primary process type — first `Terminate` and then `Launch`.

The `HealthCheck` process works in conjunction with the `ReplaceUnhealthly` process to provide health check functionality. If you suspend either the `Launch` or the `Terminate` process, the `ReplaceUnhealthy` process will not function properly.

The `ScheduledActions` process type performs scheduled actions that you create with either the PutScheduledUpdateGroupAction API action or the `as-put-scheduled-update-group-action` command. Scheduled actions often involve launching new instances or terminating existing instances. If you suspend either the `Launch` or the `Terminate` process, your scheduled actions might not function as expected.

Auto Scaling might, at times, suspend processes for Auto Scaling groups that repeatedly fail to launch instances. This is known as an administrative suspension, and most commonly applies to Auto Scaling groups that have no running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time in launching any instances.

Auto Scaling allows you to resume both, processes suspended for administrative reasons and processes suspended following a suspension of `Launch` or `Terminate`.

The following procedures walk you through the process of suspending all scaling activities on your Auto Scaling group to investigate a configuration problem with your Amazon EC2 application. After you conclude the investigation, you can resume scaling activities on the Auto Scaling group.

For

You can suspend and then resume the scaling process on your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. If you are planning on using the CLI, be sure you have installed the tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14). For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

Topics

Using the Command Line Interface

This example assumes that you have an Amazon EC2 application running within a single Auto Scaling group named `MyAutoScalingGroup`.

To suspend and resume processes on an Auto Scaling group

1. Use the `as-suspend-processes` command by specifying the following value:

   - Auto Scaling group name = `MyAutoScalingGroup`

   Your command should look similar to the following example:

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

   Auto Scaling returns the following:

```
OK-Processes Suspended
```

2. After concluding your investigation, use the `as-resume-processes` command by specifying the following value:

   - Auto Scaling group name = `MyAutoScalingGroup`

   Your command should look similar to the following example:

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

   Auto Scaling returns the following:

```
OK-Processes Resumed
```

Your Amazon EC2 application has resumed normal Auto Scaling activities.

Using the Query API

This example assumes that you have an Amazon EC2 application running within a single Auto Scaling group named `MyAutoScalingGroup`.

To suspend and then resume scaling processes on an Auto Scaling group

1. Call the `SuspendProcesses` action with the following parameters:

   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`

2. After concluding your investigation, call the `ResumeProcesses` with the following parameters:

   - *`AutoScalingGroupName`* = `MyAutoScalingGroup`

Your Amazon EC2 application resumes normal Auto Scaling activities.

Shut Down Your Auto Scaling Process

The Auto Scaling process can be completely shut down by following these steps:

- Update your Auto Scaling group by specifying 0 for your maximum, minimum, and desired (if defined) number of instances.
- Delete the Auto Scaling group.
- [optional] Delete the launch configuration.
- [optional] Delete the load balancer.
- [optional] Delete the CloudWatch alarms.

Topics

The following sections step you through the process of completely shutting down the Auto Scaling process for your Auto Scaling group.

You can shut down the Auto Scaling process for your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. If you are planning to use the CLI, be sure to install the tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14). If you have used Elastic Load Balancing and CloudWatch alarms with your Auto Scaling group and are planning to shut down your load balancer and the alarms, make sure to install the command line tools for those services as well. For information on Elastic Load Balancing CLI tools, see Installing the Command Line Interface. For information on CloudWatch CLI tools, see Command Line Tools. For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

Using the Command Line Interface

You can use CLI to completely shut down the scaling process for your Auto Scaling group.

Delete Your Auto Scaling Group

You can delete your Auto Scaling group if the group has no running instances. In order to ensure that your Auto Scaling group has no running instances, update your Auto Scaling group by specifying the minimum size and the maximum size as zero instances.

To delete your Auto Scaling group

1. **Update your Auto Scaling group**

   a. Use the `as-update-auto-scaling-group` command and specify the following values:

   - Auto Scaling group name = `my-test-asg`
   - Max size = `0`
   - Min size = `0`

   Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg --max-size 0 --min-size 0
```

b. You should get a confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

2. **Verify that your Auto Scaling group has no running instances**

   a. Use the `as-describe-auto-scaling-groups` command and specify the following values:

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

   Your command should look similar to the following example:

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

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

   After the termination process completes, you should get a confirmation like the following example:

```
AUTO-SCALING-GROUP  GROUP-NAME    LAUNCH-CONFIG  AVAILABILITY-ZONES
LOAD-BALANCERS       MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-
POLICIES
AUTO-SCALING-GROUP  my-test-asg  my-test-lc     us-east-1a          my-
test-asg-loadbalancer  0         0         0                 Default
```

3. **Delete your Auto Scaling group**

   a. Use the `as-delete-auto-scaling-group` command and specify the following values:

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

   Your command should look similar to the following example:

```
as-delete-auto-scaling-group my-test-asg
```

   b. Auto Scaling returns the following question:

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

   c. Enter y to delete the Auto Scaling group.
   d. Auto Scaling returns the following response:

```
OK-Deleted AutoScalingGroup
```

Delete the Launch Configuration Associated with Your Auto Scaling Group [optional]

Skip this step if you are planning on using the launch configuration later to launch Auto Scaling groups.

To delete launch configuration associated with your Auto Scaling group

1. Use the `as-delete-launch-config` command and specify the following value:

   - Launch configuration name = `my-test-lc`

   Your command should look similar to the following example:

   ```
   as-delete-launch-config my-test-lc
   ```

2. Auto Scaling returns the following:

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

3. Enter `y` to delete the launch configuration.
4. Auto Scaling returns the following:

   ```
   OK-Deleted launch configuration
   ```

Delete Load Balancer [optional]

Skip this step if your Auto Scaling group is not registered with your Elastic Load Balancing load balancer or you do not wish to delete your load balancer.

To delete load balancer

1. Use the Elastic Load Balancing command `elb-delete-lbs` and specify the following value:

   - Load Balancer name = `my-test-asg-loadbalancer`

   Your command should look similar to the following example:

   ```
   elb-delete-lb my-test-asg-loadbalancer
   ```

2. Elastic Load Balancing returns the following:

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

3. Enter `y` to delete the load balancer.
4. Elastic Load Balancing returns the following:

```
OK-Deleting LoadBalancer
```

Delete CloudWatch Alarms [optional]

Skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you do not wish to delete the CloudWatch alarms.

To delete CloudWatch alarms

1. Use the CloudWatch command `mon-delete-alarms` and specify the following value:

   - Alarm names = *`AddCapacity, RemoveCapacity`*

   Your command should look similar to the following example:

   ```
   mon-delete-alarms AddCapacity, RemoveCapacity
   ```

2. CloudWatch returns the following:

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

3. Enter `y` to delete the alarms.
4. You should get confirmation similar to the following example:

   ```
   OK-Deleting Alarms
   ```

Using the Query API

You can use the Query API to completely shut down the scaling process for your Auto Scaling group.

Delete Your Auto Scaling Group

You can delete your Auto Scaling group if the group has no running instances. In order to ensure that your Auto Scaling group has no running instances, update your Auto Scaling group by specifying the minimum size and the maximum size as zero instances.

To delete your Auto Scaling group

1. Update your Auto Scaling group.

   a. Call the `UpdateAutoScalingGroup` action and specify the following parameters:

      - AutoScalingGroupName = `my-test-asg`
      - MaxSize = `0`
      - MinSize = `0`

      Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&MinSize=0
&MaxSize=0
&Version=2011-01-01
&Action=UpdateAutoScalingGroup
&AUTHPARAM
```

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>0d361eb7-9665-11e2-80ec-8d4dEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

2. Verify that your Auto Scaling group has no running instances.

   a. Call the `DescribeAutoScalingGroups` action and specify the following parameter:

      - AutoScalingGroupName = `my-test-asg`

      Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupNames.member.1=my-
test-asg
&MaxRecords=20
&Version=2011-01-01
&Action=DescribeAutoScalingGroups
&AUTHPARAMS
```

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

      After the termination process completes, you should get a confirmation like the following example:

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-03-17T03:54:14.210Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances/>
        <DesiredCapacity>0</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1a</member>
```

```
        </AvailabilityZones>
        <LoadBalancerNames>
          <member>my-test-asg-loadbalancer</member>
        </LoadBalancerNames>
        <MinSize>0</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:23639c1d-6703-4d8e-bd04-
1428a16e0770:autoScalingGroupName/my-test-asg</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>0</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>57d72661-9664-11e2-b1f1-2f998EXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

3. Delete your Auto Scaling group.

   a. Call the `DeleteAutoScalingGroup` action and specify the following parameter:

      - AutoScalingGroupName = `my-test-asg`

      Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg
&ForceDelete=false
&Version=2011-01-01
&Action=DeleteAutoScalingGroup
&AUTHPARAM
```

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

```
<DeleteAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>70a76d42-9665-11e2-9fdf-211deEXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteAutoScalingGroupResponse>
```

Delete Launch Configuration Associated With Your Auto Scaling Group [optional]

Skip this step if you are planning on using the launch configuration later to launch Auto Scaling groups.

To delete launch configuration associated with your Auto Scaling group

1. Call the `DeleteLaunchConfiguration` action and specify the following parameter:

   - LaunchConfigurationName = `my-test-lc`

   Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?LaunchConfigurationName=my-test-lc
&Version=2011-01-01
&Action=DeleteLaunchConfiguration
&AUTHPARAM
```

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

```
<DeleteLaunchConfigurationResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>7347261f-97df-11e2-8756-35eEXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteLaunchConfigurationResponse>
```

Delete Load Balancer [optional]

Skip this step if your Auto Scaling group is not registered with Elastic Load Balancing load balancer or you do not wish to delete your load balancer.

To delete the load balancer

1. Call the `DeleteLoadBalancer` Elastic Load Balancing action and specify the following parameter:

   - LoadBalancerName = `my-test-asg-loadbalancer`

   Your request should look similar to the following example:

```
https://elasticloadbalancing.amazonaws.com/?LoadBalancerName=my-test-asg-
loadbalancer
&Version=2012-06-01
&Action=DeleteLoadBalancer
&AUTHPARAMS
```

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

```
<DeleteLoadBalancerResponse xmlns="http://elasticloadbalancing.amazon
aws.com/doc/2012-06-01/">
```

```
  <DeleteLoadBalancerResult/>
  <ResponseMetadata>
    <RequestId>806af50e-97ee-11e2-b66d-8133EXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteLoadBalancerResponse>
```

Delete CloudWatch Alarms [optional]

Skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you do not wish to delete the CloudWatch alarms.

To delete CloudWatch alarms

1. Call the `DeleteAlarms` CloudWatch action and specify the following parameter:

   - AlarmNames.member.1 = *`AddCapacity`*
   - AlarmNames.member.2 = *`RemoveCapacity`*

   Your request should look similar to the following example:

```
https://monitoring.us-east-1.amazonaws.com/?AlarmNames.member.1=AddCapacity
&AlarmNames.member.2=RemoveCapacity
&Version=2010-08-01
&Action=DeleteAlarms
&AUTHPARAMS
```

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

```
<DeleteAlarmsResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-
01/">
  <ResponseMetadata>
    <RequestId>66c9f789-adc9-11e2-981d-c7104EXAMPLE</RequestId>
  </ResponseMetadata>
</DeleteAlarmsResponse>
```

Working With Auto Scaling

This section discusses some common scenarios for using Auto Scaling. Each scenario describes the situation, identifies the pieces you need and why, lists the tools you need and how to use them, and shows you the steps to complete the task.

If you aren't already acquainted with the basic concepts behind Auto Scaling, see What is Auto Scaling? (p. 1).

You can access Auto Scaling by downloading and installing the Auto Scaling command line interface (CLI), by creating a query request with the query API, or by using the AWS SDKs. The procedures in this section include instructions for the command line interface and the Query API. For information on downloading and using the Auto Scaling interfaces, see Get Started With Auto Scaling Interfaces (p. 12).

In all our example procedures, we assume that your instances are in the US East (Northern Virginia) Region. If your instances are in a different region, you must specify the region where your instances reside. For example, if your instances are in Europe, you must specify the *eu-west-1* Region by using the `--region eu-west-1` parameter or setting the `EC2_REGION` environment variable. For information on specifying a region, see How to Change the Region (p. 17).

The walkthroughs in this section lead you through the following scenarios:

- Load Balance Your Auto Scaling Group (p. 126)

  This scenario walks you through the processes for registering your Elastic Load Balancing load balancer with your Auto Scaling group, adding an Elastic Load Balancing health check, and expanding your auto-scaled and load-balanced application to an additional Availability Zone.
- Use Amazon SQS Queues to Determine When to Auto Scale (p. 140)

  This scenario walks you through the process of creating Auto Scaling policies for configuring your Auto Scaling group to scale based on number of messages in an Amazon Simple Queue Service (Amazon SQS) queue.
- Get Email Notifications When Your Auto Scaling Group Changes (p. 156)

  This scenario walks you through the process of setting up Amazon Simple Notification Service (Amazon SNS) to send email notifications to your email address with information about the instances in your Auto Scaling group.
- Launch Spot Instances in Your Auto Scaling Group (p. 161)

  This scenario walks you through the process of launching Spot Instances in your Auto Scaling group.
- Control Access to Auto Scaling Resources (p. 173)

This scenario walks you through the process to create users and user groups under your organization's AWS account, to grant or deny access to your resources.

- Launch Auto Scaling Instances with IAM Role (p. 176)

  This scenario walks you through the process of launching EC2 instances in your Auto Scaling group with an AWS Identity and Access management (IAM) role. Instances launched with an IAM role will automatically have AWS security credentials available.

- Monitor Your Auto Scaling Instances (p. 179)

  Amazon CloudWatch offers basic or detailed monitoring of the instances in your Auto Scaling group. This scenario walks you through the process for enabling either detailed monitoring or basic monitoring of the instances.

Load Balance Your Auto Scaling Group

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

This topic shows you how you can use Elastic Load Balancing to route traffic to Amazon EC2 instances in your Auto Scaling group. If you aren't already acquainted with basic Auto Scaling concepts, see What is Auto Scaling? (p. 1). For information on Elastic Load Balancing, see What Is Elastic Load Balancing?

Elastic Load Balancing uses load balancers to monitor traffic and handle requests that come through the Internet. To use Elastic Load Balancing with your Auto Scaling group, you first have to create a load balancer and then you need to register your Auto Scaling group with the load balancer. Your load balancer acts as a single point of contact for all incoming traffic. You can register multiple load balancers with a single Auto Scaling group. For information on registering your load balancer with your Auto Scaling group, see Set Up an Auto-Scaled and Load-Balanced Application (p. 127).

Elastic Load Balancing sends data about your load balancers and EC2 instances to Amazon CloudWatch. CloudWatch collects the data and presents it as readable, near-time metrics. After registering the load balancer with your Auto Scaling group, you can configure your Auto Scaling group to use Elastic Load Balancing metrics (such as request latency or request count) to auto scale your application. For information on Elastic Load Balancing metrics, see Monitor Your Load Balancer Using Amazon CloudWatch. For information on using CloudWatch metrics to auto scale, see Scale Based on Demand (p. 48).

By default, the Auto Scaling group determines the health state of each instance by periodically checking the results of Amazon EC2 instance status checks. Elastic Load Balancing also performs health checks on the EC2 instances that are registered with the load balancer. After you've registered your Auto Scaling group with a load balancer, you can choose to use the results of the Elastic Load Balancing health check in addition to the EC2 instance status checks to determine the health of the EC2 instances in your Auto Scaling group. For information on adding an Elastic Load Balancing health check, see Add an Elastic Load Balancing Health Check to your Auto Scaling Group (p. 133).

You can take advantage of the safety and reliability of geographic redundancy by spanning your Auto Scaling groups across multiple Availability Zones within a region and then setting up load balancers to distribute incoming traffic across those Availability Zones. For information on expanding your auto-scaled and load-balanced application to an additional Availability Zone, see Expand Your Auto-Scaled and Load-Balanced Application to an Additional Availability Zone (p. 134).

Set Up an Auto-Scaled and Load-Balanced Application

The following sections step you through the process of registering your Auto Scaling group with an Elastic Load Balancing load balancer to set up an auto-scaled and load-balanced application.

We will step you through the following processes:

1. [optional] How to create a launch configuration. Skip this step if you want to use your own launch configuration.
2. How to create an Auto Scaling group with a load balancer.
3. How to verify that your Auto Scaling group has been created with a load balancer

Prerequisites

Before you begin registering your load balancer with your Auto Scaling group, be sure you have completed the following prerequisites:

- Sign up for Amazon Web Services (AWS).

  If you haven't yet signed up, go to http://aws.amazon.com, click **Sign Up**, and follow the on-screen instructions.
- Follow the steps in Get Started With Elastic Load Balancing, to create a load balancer. When creating your load balancer, you can skip the step for registering your Amazon EC2 instances.

  Note
  You don't need not register your Amazon EC2 instances with your load balancer if you plan to attach your load balancer to an Auto Scaling group. Auto Scaling will launch Amazon EC2 instances when you create your Auto Scaling group and then attach the Auto Scaling group to the load balancer.
- You can use the Auto Scaling command line interface (CLI) or the Query API to set up an auto-scaled and load-balanced application. If you plan to use the CLI, be sure you have installed the tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14). For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

Topics

Using the Command Line Interface

In this walkthrough, you'll create a launch configuration `my-test-lc`, an Auto Scaling group `my-test-asg-lbs`, and register the load balancer `my-test-asg-loadbalancer` that you created created in the previous step.

Create Launch Configuration

If you'd rather use your own launch configuration, skip the following procedure.

To create the launch configuration

1. Use the `as-create-launch-config` command and specify the following values:

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

  Note
  The image ID is provided for illustration purposes only. Image IDs change over time. To obtain a list of current valid image ID, see Finding a Suitable AMI in the *Amazon Elastic Compute Cloud User Guide*.
- Instance type = `m1.small`

Your command should look similar to the following example:

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

2. You should get a confirmation like the following example:

```
OK-Created launch config
```

Create Auto Scaling Group

You can use the following procedure to create an Auto Scaling group and attach a load balancer.

To create an Auto Scaling group

1. Use the `as-create-auto-scaling-group` command and specify the following values:

   - Auto Scaling group name = `my-test-asg-lbs`
   - Launch configuration name = `my-test-lc`
   - Availability Zone = `us-east-1a,us-east-1b`
   - Load Balancer name = *`my-test-asg-loadbalancer`*
   - Max size = 6
   - Min size = 2
   - Desired capacity = 2

   Note
   The load of the incoming traffic is balanced equally across all Availability Zones enabled for your load balancer. Auto Scaling tries to launch an equivalent number of instances in each zone. As a best practice, we recommend that you specify even numbers for maximum, minimum, and desired capacity for your Auto Scaling group.

   Your command should look similar to the following example:

```
as-create-auto-scaling-group my-test-asg-lbs --launch-configuration my-test-
lc --availability-zones
us-east-1a,us-east-1b  --load-balancers my-test-asg-loadbalancer --max-size
 6 --min-size 2 --desired-capacity 2
```

2. You should get a confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Verify that Your Auto Scaling Group Launched with a Load Balancer

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

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

1. Use the `as-describe-auto-scaling-groups` command and specify the following value:

   - Auto Scaling group name = `my-test-asg-lbs`

   Your command should look similar to the following example:

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

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

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

   ```
   AUTO-SCALING-GROUP  GROUP-NAME       LAUNCH-CONFIG  AVAILABILITY-ZONES
    LOAD-BALANCERS       MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-
   POLICIES
   AUTO-SCALING-GROUP  my-test-asg-lbs  my-test-lc    us-east-1b,us-east-1a
   my-test-asg-loadbalancer  2         6         2                 Default
   INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
   INSTANCE  i-78e60b1b   us-east-1b         InService  Healthy  my-test-lc1
   INSTANCE  i-941599fe   us-east-1a         InService  Healthy  my-test-lc1
   ```

You can see that Auto Scaling has registered the load balancer `my-test-asg-loadbalancer` and launched two instances using the launch configuration `my-test-lc`.

Using the Query API

In this walkthrough, you'll create a launch configuration `my-test-lc`, an Auto Scaling group `my-test-asg-lbs`, and register the load balancer `my-test-asg-loadbalancer` created in the previous step, with the Auto Scaling group.

Create Launch Configuration

If you'd rather use your own launch configuration, skip the following procedure.

To create the launch configuration

1. Call the `CreateLaunchConfiguration` action and specify the following parameters:

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

Note
The image ID is provided for illustration purposes only. Image IDs change over time. To obtain a list of current valid image ID, see Finding a Suitable AMI in the *Amazon Elastic Compute Cloud User Guide*.

- InstanceType = `m1.small`

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?LaunchConfigurationName=my-test-lc
&ImageId=ami-514ac838
&InstanceType=m1.small
&Action=CreateLaunchConfiguration
&AUTHPARAMS
```

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

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

Create Auto Scaling Group

The following procedure walks you through process of creating an Auto Scaling group and attaching a load balancer.

To create an Auto Scaling group

1. Call the `CreateAutoScalingGroup` action and specify the following parameters:

   - AutoScalingGroupName = `my-test-asg-lbs`
   - LaunchConfigurationName = `my-test-lc`
   - AvailabilityZones.member.1 = `us-east-1a`
   - AvailabilityZones.member.2 = `us-east-1b`
   - LoadBalancerNames.member.1 = *`my-test-asg-loadbalancer`*
   - MaxSize = 6
   - MinSize = 2
   - DesiredCapacity = 2

   Note
   The load of the incoming traffic is balanced equally across all Availability Zones enabled for your load balancer. Auto Scaling tries to launch an equivalent number of instances in each zone. As a best practice, we recommend that you specify even numbers for maximum, minimum, and desired capacity for your Auto Scaling group.

   Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg-lbs
&LoadBalancerNames.member.1=my-test-asg-loadbalancer
&AvailabilityZones.member.1=us-east-1a
&AvailabilityZones.member.2=us-east-1b
&MinSize=2
&MaxSize=6
&DesiredCapacity=2
&LaunchConfigurationName=my-test-lc
&Version=2011-01-01
&Action=CreateAutoScalingGroup
&AUTHPARAMS
```

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

```
<CreateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <ResponseMetadata>
    <RequestId>00bc5d78-8eb7-11e2-95f9-c35bfEXAMPLE</RequestId>
  </ResponseMetadata>
</CreateAutoScalingGroupResponse>
```

Verify that Your Auto Scaling Group Launched with a Load Balancer

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

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

1. Call the `DescribeAutoScalingGroups` action and specify the following parameter:

   - AutoScalingGroupNames.member.1 = `my-test-asg-lbs`

   Your request should look similar to the following example:

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

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

```
<DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg-lbs</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
```

```
        <CreatedTime>2012-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1b</AvailabilityZone>
            <InstanceId>i-78e60b1b</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-941599fe</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>2</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1b</member>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames>
          <member>my-test-asg-loadbalancer</member>
        </LoadBalancerNames>
        <MinSize>2</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:e8b084c9-8cad-444a-8381-c97b778e0fc0:auto
ScalingGroupName/my-test-asg-l
bs</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>6</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>95fdfeb8-aa76-11e2-81e1-750aaEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribeAutoScalingGroupsResponse>
```

You can see that Auto Scaling registered the load balancer `my-test-asg-loadbalancer` and launched two instances using the launch configuration `my-test-lc`. The newly launched instances are healthy, and running (`InService`).

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

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

For information about Amazon EC2 instance status checks, see Monitor Instances With Status Checks. For information about Elastic Load Balancing health checks, see Health Check.

This topic shows you how to add an Elastic Load Balancing health check to your Auto Scaling group. We assume that you have created a load balancer and have registered the load balancer with your Auto Scaling group. If you have not registered the load balancer with your Auto Scaling group, see Set Up an Auto-Scaled and Load-Balanced Application (p. 127).

Auto Scaling marks an instance unhealthy if the calls to the Amazon EC2 action DescribeInstanceStatus returns any other state other than `running`, the system status shows `impaired`, or the calls to Elastic Load Balancing action DescribeInstanceHealth returns `OutOfService` in the instance state field.

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

You can add an Elastic Load Balancing health check to your Auto Scaling group using the Auto Scaling command line interface (CLI) or the Query API. If you plan to use the CLI, be sure you have installed the tools. For information on installing the command line interface, see Install the Command Line Interface (p. 14). For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

Topics

Using the Command Line Interface

To add an Elastic Load Balancing health check, use the `as-update-auto-scaling-group` command and specify the following values:

- Auto Scaling group name = *`my-test-asg-lbs`*
- Health check type = `ELB`
- Health check grace period = *`300`*

> **Note**
> Frequently, new instances need to briefly warm up before they can pass a health check. To provide ample warm-up time, set the health check grace period of the group to match the expected startup period of your application.

Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg-lbs –-health-check-type ELB  –-grace-
period 300
```

You should get a confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

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

Using the Query API

To add an Elastic Load Balancing health check, call the UpdateAutoScalingGroup action by specifying the following parameters:

- AutoScalingGroupName = *my-test-asg-lbs*
- HealthCheckType = `ELB`
- HealthCheckGracePeriod = *300*

> **Note**
> Frequently, new instances need to briefly warm up before they can pass a health check. To provide ample warm-up time, set the health check grace period of the group to match the expected startup period of your application.

Your request should look similar to the following example:

```
https://autoscaling.amazonaws.com/?HealthCheckType=ELB
&HealthCheckGracePeriod=300
&AutoScalingGroupName=my-test-asg-lbs
&Version=2011-01-01
&Action=UpdateAutoScalingGroup
&AUTHPARAMS
```

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

```
<UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
01-01/">
  <ResponseMetadata>
    <RequestId>adafead0-ab8a-11e2-ba13-ab0ccEXAMPLE</RequestId>
  </ResponseMetadata>
</UpdateAutoScalingGroupResponse>
```

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

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

When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the designated Availability Zones for your Auto Scaling group. Auto Scaling does this by attempting to launch new instances in the

Availability Zone with the fewest instances. If the attempt fails, however, Auto Scaling will attempt to launch in other zones until it succeeds.

An Auto Scaling group can contain EC2 instances that come from one or more Availability Zones within the same region. However, an Auto Scaling group cannot span multiple regions. For information about the regions and Availability Zones supported by Auto Scaling, see Regions and Endpoints.

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

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

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

You can use the Auto Scaling command line interface along with the Elastic Load Balancing command line interface (CLI) to add an Availability Zone to your auto-scaled and load-balanced application. You can also use the Query API. If you plan on using the CLIs, be sure you have installed the tools. For information on installing the Auto Scaling CLI, see Install the Command Line Interface (p. 14). For information on installing the Elastic Load Balancing CLI, see Installing the Command Line Interface. For information on creating a query request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

Topics

Using the Command Line Interface

In this example, you learn how to expand the availability of your application to an additional Availability Zone, `us-east-1c`.

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

1. Use the `as-update-auto-scaling-group` command and specify the following values:

   - Auto Scaling group name = `my-test-asg-lbs`
   - Availability Zones = `us-east-1a, us-east-1b, us-east-1c`
   - min size = `3`

   Your command should look similar to the following example:

```
as-update-auto-scaling-group my-test-asg-lbs --availability-zones us-east-
1a, us-east-1b, us-east-1c --min-size 3
```

You should get a confirmation like the following example:

```
OK-Updated AutoScalingGroup
```

2. Use the `as-describe-auto-scaling-groups` command and specify the following value:

   - Auto Scaling group name = *`my-test-asg-lbs`*

   Your command should look similar to the following example:

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

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

   ```
   AUTO-SCALING-GROUP  GROUP-NAME       LAUNCH-CONFIG  AVAILABILITY-ZONES
   LOAD-BALANCERS  MIN-SIZE  MAX-SIZE  DESIRED-CAPACITY  TERMINATION-POLICIES
   AUTO-SCALING-GROUP  my-test-asg-lbs  my-test-lc    us-east-1c,us-east-1b,us-
   east-1a  my-test-asg-loadbalancer  3         6         3     Default

   INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
   INSTANCE  i-78e60b1b   us-east-1b         InService  Healthy  my-test-lc
   INSTANCE  i-dd4b2eb2   us-east-1c         InService  Healthy  my-test-lc
   INSTANCE  i-48a1cf29   us-east-1a         InService  Healthy  my-test-lc
   ```

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

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

3. Use the Elastic Load Balancing `elb-enable-zones-for-lb` command by specifying the following values:

   - Load balancer name = *`my-test-asg-loadbalancer`*
   - Availability Zone = `us-east-1c`

   Your command should look similar to the following example:

   ```
   elb-enable-zones-for-lb  my-test-asg-loadbalancer   --availability-zones
   us-east-1c --headers
   ```

   Elastic Load Balancing responds with a list of Availability Zones enabled for the load balancer. The information you get should be similar to the following example:

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

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

Using the Query API

In this example, you learn how to expand the availability of your application to an additional Availability Zone, `us-east-1c`.

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

1. Call the UpdateAutoScalingGroup action and specify the following parameters:

   - AutoScalingGroupName = `my-test-asg`
   - AvailabilityZones.member.1 = `us-east-1a`
   - AvailabilityZones.member.2 = `us-east-1b`
   - AvailabilityZones.member.3 = `us-east-1c`
   - MinSize = `3`

   Your request should look similar to the following example:

   ```
   https://autoscaling.amazonaws.com/?AutoScalingGroupName=my-test-asg-lbs
   &AvailabilityZones.member.1=us-east-1a
   &AvailabilityZones.member.2=us-east-1b
   &AvailabilityZones.member.3=us-east-1c
   &MinSize=3
   &Version=2011-01-01
   &Action=UpdateAutoScalingGroup
   &AUTHPARAMS
   ```

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

   ```
   <UpdateAutoScalingGroupResponse xmlns="http://autoscaling.amazon
   aws.com/doc/2011-01-01/">
     <ResponseMetadata>
       <RequestId>325f71b3-ac46-11e2-9cae-61a8EXAMPLE</RequestId>
     </ResponseMetadata>
   </UpdateAutoScalingGroupResponse>
   ```

2. Call the `DescribeAutoScalingGroups` action and specify the following parameter:

   - AutoScalingGroupNames.member.1 = `my-test-asg-lbs`

   Your request should look similar to the following example:

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

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

   ```
   <DescribeAutoScalingGroupsResponse xmlns="http://autoscaling.amazon
   aws.com/doc/2011-01-01/">
   ```

```
  <DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg-lbs</AutoScalingGroupName>
        <HealthCheckType>ELB</HealthCheckType>
        <CreatedTime>2013-04-21T11:12:17.795Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1a</AvailabilityZone>
            <InstanceId>i-44a7b627</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1b</AvailabilityZone>
            <InstanceId>i-c34f20a3</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1c</AvailabilityZone>
            <InstanceId>i-98562cf1</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>3</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1c</member>
          <member>us-east-1b</member>
          <member>us-east-1a</member>
        </AvailabilityZones>
        <LoadBalancerNames>
          <member>my-test-asg-loadbalancer</member>
        </LoadBalancerNames>
        <MinSize>3</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>300</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:e8b084c9-8cad-444a-8381-c97b778e0fc0:auto
ScalingGroupName/my-test-asg-lbs</AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>6</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>084332c2-ac57-11e2-b92b-45061efc08bd</RequestId>
```

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

Check the status of the instances in the `<LifecycleState>` field of all three instances. When the status of each of the new instances in the new Availability Zone appears as `InService`, indicating that the instances are now recognized by the Elastic Load Balancing and ready, proceed to the next step.

Note
When you call `EnableAvailabilityZonesForLoadBalancer`, the load balancer begins to route traffic equally among all the enabled Availability Zones.

4. Call the EnableAvailabilityZonesForLoadBalancer action by specifying the following parameters:

   - *`AvailabilityZones`* = `us-east-1c`
   - *`LoadBalancerName`* = `my-test-asg-loadbalancer`

   Your request should look similar to the following example:

```
https://elasticloadbalancing.amazonaws.com/?AvailabilityZones.member.1=us-
east-1c
&LoadBalancerName=my-test-asg-loadbalancer
&Version=2012-06-01
&Action=EnableAvalabilityZonesForLoadBalancer
&AUTHPARAMS
```

5. The response includes a list of Availability Zones enabled for the load balancer. The information you get should be similar to the following example:

```
<EnableAvailabilityZonesForLoadBalancerResponse xmlns="http://elasticload
balancng.amazonaws.com/doc/2012-06-01/">
  <EnableAvailabilityZonesForLoadBalancerResult>
    <AvailabilityZones>
      <member>us-east-1c</member>
      <member>us-east-1b</member>
      <member>us-east-1a</member>
    </AvailabilityZones>
  </EnableAvailabilityZonesForLoadBalancerResult>
  <ResponseMetadata>
    <RequestId>1ae1f97a-ac59-11e2-ac73-fffddEXAMPLE</RequestId>
  </ResponseMetadata>
</EnableAvailabilityZonesForLoadBalancerResponse>
```

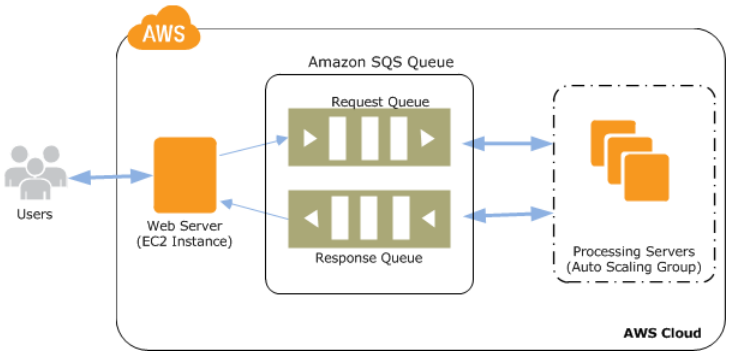
You have load-balanced your Amazon EC2 application across three Availability Zones and have auto-scaled it in each zone.

Use Amazon SQS Queues to Determine When to Auto Scale

Amazon Simple Queue Service (Amazon SQS) is a scalable message queuing system that stores messages as they travel between various components of your application architecture. Amazon SQS enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component. A queue is a temporary repository for messages that are awaiting processing. When you use Amazon SQS queues in your architecture, you introduce an interface between the various components, which allows the components to be decoupled from one another. The independent components are easy to scale. This topic shows you how you can use SQS queues to establish thresholds that Auto Scaling can use to scale the components in your architecture.

For more information on Amazon SQS, see What is Amazon Simple Queue Service? If you aren't already acquainted with the basic concepts behind Auto Scaling, see What is Auto Scaling? (p. 1)

We'll use an example to illustrate how Auto Scaling and Amazon SQS work together. Let's say that you have a web application that receives orders from customers. The application, places the orders in a queue until they are picked up for processing, processes the orders, and then sends the processed orders back to the customer. In this example, you have an application running on an EC2 instance that receives the orders and sends them to the Amazon SQS queue. The SQS queue stores the orders until they are ready to be processed. You have configured an Auto Scaling group to receive the orders from the queue, process the orders, and send them back to the queue. For this example, the capacity in your Auto Scaling group is configured to handle a normal order load. The following diagram illustrates the architecture of this example.



This architecture works well if your order levels remain the same at all times. What happens if your order levels change? You would need to launch additional EC2 instances when you start getting more orders than you normally do, and later you would terminate those additional EC2 instances when the orders return to normal levels. If your order levels increase and decrease on a predictable schedule, you can specify the time and date to perform these scaling activities. For more information, see Scale Based on a Schedule (p. 65).

What happens if you can’t predict when you will get more orders? In that case you will have to identify the conditions that determine the increasing and decreasing order loads. Then you need to tell Auto Scaling to launch or terminate EC2 instances when those conditions are met. Queues provide a convenient mechanism to determine the load on an application. You can use the length of the queue (number of messages available for retrieval from the queue) to determine the load. Because each message in the queue represents a request from a user, measuring the length of the queue is a fair approximation of the

load on the application. By performing load tests on the queue, you can determine the optimal length of the queue (the length at which you have the required number of EC2 instances running to cover the demand). At any time, if the current length of the queue exceeds the optimal length, then you should start additional EC2 instances. Likewise, if the current length falls below the optimal length, then it's time to terminate the additional EC2 instances.

Amazon CloudWatch integrates with Amazon SQS to collect, view, and analyze metrics from Amazon SQS queues. You can use the metrics sent by SQS to determine the length of the SQS queue at any point in time. For a list of all the metrics that Amazon SQS sends to Amazon CloudWatch, see Amazon SQS Metrics.

The following sections step you through the process of creating Auto Scaling policies for configuring your Auto Scaling group to scale based on the number of messages in your Amazon SQS queue. Policies tell Auto Scaling what to do when the specified conditions occur. For information on using Auto Scaling policies, see Scale Based on Demand (p. 48). You can create Auto Scaling policies using the Auto Scaling command line interface (CLI) or the Query API.

Create Auto Scaling Policies and CloudWatch Alarms

Topics

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

1. How to create an Auto Scaling policy to scale out (launch instances).
2. How to create another Auto Scaling policy to scale in (terminate instances).
3. How to create an Amazon CloudWatch alarm to watch the Amazon Simple Queue Service (Amazon SQS) metric `ApproximateNumberOfMessagesVisible` and then attach either the Auto Scaling policy for scaling in or for scaling out.
4. How to verify that your Auto Scaling policies and CloudWatch alarms have been created.
5. How to validate that the scaling activity is taking place when the specified condition occurs.

Prerequisites

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

- Sign up for Amazon Web Services (AWS) Services.

  If you haven't yet signed up, go to http://aws.amazon.com, click **Sign Up**, and follow the on-screen instructions.
- Follow the steps in Working with Amazon SQS to create a queue, confirm that the queue exists, and send and receive messages to and from your queue.
- Create an Auto Scaling group. For more information, see Basic Auto Scaling Configuration (p. 32). Make a note of the instance ID of the newly launched instance. You'll need this later in the walkthrough.
- Configure the EC2 instance in your Auto Scaling group to receive, send, and process messages using Amazon SQS.

Using the Command Line Interface

In this walkthrough, you'll create two scaling policies, `my-sqs-scaleout-policy`, which increases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance, and `my-sqs-scalein-policy`, which decreases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance. You can optionally specify a custom `cooldown` period. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. For more information, see Cooldown Period (p. 7).

Create Scaling Policies

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

To create scaling policies

1. Use the `as-put-scaling-policy` command and specify the following values:

   - Policy name = `as-sqs-scaleout-policy`
   - Auto Scaling group name = `my-test-asg`
   - Adjustment = `1`
   - Adjustment type = `ChangeInCapacity`
   - [optional] Cooldown = *`cooldown period`*

   Your command should look similar to the following example:

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

   Note
   If you are running Windows, you must use quotation marks when specifying the adjustment value, for example `"--adjustment=-1"`.
   No Auto Scaling name, including a policy name, can contain the colon (:) character because colons serve as delimiters in Amazon Resource Names (ARNs).

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

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

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

3. Use the `as-put-scaling-policy` command to create another policy by specifying the following values:

   - Policy name = `as-sqs-scalein-policy`
   - Auto Scaling group name = `my-test-asg`
   - Adjustment = `-1`
   - Adjustment type = `ChangeIn Capacity`
   - [optional] Cooldown = *`cooldown period`*

   Your command should look similar to the following example:

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

4. Auto Scaling returns the ARN for the policy.

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

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

Create CloudWatch Alarms

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

Before you begin, be sure you have installed the Amazon CloudWatch command line interface. For more information, see Command Line Tools.

To create CloudWatch alarms

1. Use the Amazon CloudWatch command mon-put-metric-alarm to create an alarm to increase the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) increases to three and remains at three or more for a period of one minute.

   Note
   In this example, the values for `Threshold` and `Period` are kept low to invoke and test the scaling activity. You can change the values after validating that the scaling activity is taking place.

   Specify the following values:

   - Alarm name = `AddCapacityToProcessQueue`
   - Metric name = `ApproximateNumberOfMessagesVisible`
   - Namespace = `"AWS/SQS"`
   - Statistic = `Average`
   - Period = `60`
   - Threshold = `3`
   - Comparison operator = `GreaterThanOrEqualToThreshold`
   - Dimensions = `"QueueName=`*`MyQueue`*`"`
   - Evaluation periods = `2`
   - Alarm action =
     arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy

   Your command should look like the following example:

```
prompt>mon-put-metric-alarm --alarm-name AddCapacityToProcessQueue  --metric-
name ApproximateNumberOfMessagesVisible --namespace “AWS/SQS”
--statistic Average --period 60 --threshold 3 --comparison-operator Greater
```

```
ThanOrEqualToThreshold --dimensions “QueueName=MyQueue”
--evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:803981987763:scalingPolicy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScal
ingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy
```

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

```
OK-Created Alarm
```

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

   - Alarm name = `RemoveCapacityFromTheProcessQueue`
   - Metric name = `ApproximateNumberOfMessagesVisible`
   - Namespace = `"AWS/SQS"`
   - Statistic = `Average`
   - Period = `60`
   - Threshold = `1`
   - Comparison operator = `LessThanOrEqualToThreshold`
   - Dimensions = `"QueueName=MyQueue"`
   - Evaluation periods = `2`
   - Alarm action = arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy

   Your command should look like the following example:

```
mon-put-metric-alarm --alarm-name RemoveCapacityFromTheProcessQueue --metric-
name ApproximateNumberOfMessagesVisible --namespace “AWS/SQS”
 --statistic Average  --period 60  --threshold 1  --comparison-operator
LessThanOrEqualToThreshold  --dimensions “QueueName=MyQueue”
 --evaluation-periods 2 --alarm-actions arn:aws:autoscaling:us-east-
1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScal
ingGroupName/my-test-asg:policyName/my-sqs-scalein-policy
```

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

```
OK-Created Alarm
```

Verify Your Scaling Policies and CloudWatch Alarms

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

To verify your CloudWatch alarms

1. Use the Amazon CloudWatch command mon-describe-alarms and specify the following values:

- Alarm names = `AddCapacityToProcessQueue RemoveCapacityFromTheProcessQueue`

Your command should look like the following example:

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

2. The command returns the following information:

```
ALARM    STATE ALARM_ACTIONS  NAMESPACE  METRIC_NAME    PERIOD  STATISTIC
 EVAL_PERIODS  COMPARISON     THRESHOLD
RemoveCapacityFromProcessQueue OK arn:aws:autoscaling...policyName/my-sqs-
scalein-policy AWS/SQS ApproximateNumberOfMessagesVisible 60 Average 5
LessThanOrEqualToThreshold 1
AddCapacityToProcessQueue OK arn:aws:autoscaling...:policyName/my-sqs-sca
leout-policy AWS/SQS ApproximateNumberOfMessagesVisible 60 Average 5 Great
erThanOrEqualToThreshold 3
```

To verify your scaling policies

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

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

2. The command returns the following information:

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

Test for Scaling Out and Scaling In

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

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

1. Follow the steps in Sending a Message to add messages to the SQS queue you created earlier for this example. Make sure you have at least three messages in the queue.
2. It takes a few minutes for the SQS queue metric `ApproximateNumberOfmessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to launch one EC2 instance.
3. Use the `as-describe-auto-scaling-groups` command and specify the following value:

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

   Your command should look similar to the following example:

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

   The command returns the following information:

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

   Your Auto Scaling group has launched an additional EC2 instance.

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

1. Follow the steps in Deleting a Message to remove messages from the SQS queue. Make sure you have no more than one message in the queue.
2. It takes a few minutes for the SQS queue metric `ApproximateNumberOfmessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to terminate one EC2 instance.
3. Use `as-describe-auto-scaling-groups` command and specify the following value:

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

   Your command should look similar to the following example:

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

   The command returns the following information:

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

   Your Auto Scaling group has terminated the additional EC2 instance.

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

Using the Query API

If you aren't already acquainted with the procedure for creating a Query API request, see Use Query Requests to Call Auto Scaling APIs (p. 20).

In this walkthrough, you'll create two scaling policies, `my-sqs-scaleout-policy`, which increases the capacity of the `my-test-asg` Auto Scaling group by one EC2 instance, and `my-sqs-scalein-policy`, which decreases the capacity of `my-test-asg` Auto Scaling group by one EC2 instance. You can optionally specify a custom `cooldown` period. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. For more information, see Cooldown Period (p. 7).

Create Scaling Policies

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

To create scaling policies

1. Call the `PutScalingPolicy` action and specify the following parameters:

   - PolicyName = `as-sqs-scaleout-policy`
   - AutoScalingGroupName = `my-test-asg`
   - ScalingAdjustment = `1`
   - AdjustmentType = `ChangeInCapacity`
   - [optional] Cooldown = *`cooldown period`*

   Your request should look similar to the following example:

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

   Note
   If you are running Windows, you must use quotation marks when specifying the adjustment value, for example `"--adjustment=-1"`.
   No Auto Scaling name, including a policy name, can contain the colon (:) character because colons serve as delimiters in Amazon Resource Names (ARNs).

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

   ```
   <PutScalingPolicyResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
   01-01/">
   <PutScalingPolicyResult>
   <PolicyARN> arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-
   9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-
   ```

```
sqs-scaleout-policy </PolicyARN>
</PutScalingPolicyResult>
<ResponseMetadata><RequestId>2b8415c9-657d-11e2-b53e-5db54EXAMPLE</RequestId>
</ResponseMetadata>
</PutScalingPolicyResponse>
```

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

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

   - PolicyName = `as-sqs-scalein-policy`
   - AutoScalingGroupName = `my-test-asg`
   - ScalingAdjustment = `-1`
   - AdjustmentType = `ChangeInCapacity`
   - [optional] Cooldown = *`cooldown period`*

   Your request should look similar to the following example:

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

4. Auto Scaling returns the ARN for the policy.

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

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

Create CloudWatch Alarms

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

To create CloudWatch alarms

1. Use the CloudWatch action `PutMetricAlarm` to create an alarm to increase the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) increases to three and remains at three or more for a period of one minute.

Note
In this example, the values for `Threshold` and `Period` are kept low to invoke the conditions for scaling. You can change the values after validating that the scaling activity is taking place.

Specify the following values:

- AlarmName = `AddCapacityToProcessQueue`
- MetricName = `ApproximateNumberOfMessagesVisible`
- Namespace = `"AWS/SQS"`
- Statistic = `Average`
- Period = `60`
- Threshold = `3`
- ComparisonOperator = `GreaterThanOrEqualToThreshold`
- Dimensions.member.1 = `"QueueName=`*`MyQueue`*`"`
- EvaluationPeriods = `2`
- AlarmActions.member.1 = arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-asg:policyName/my-sqs-scaleout-policy

Your request should look like the following example:

```
https://monitoring.us-east-1.amazonaws.com/?Statistic=Average
&Threshold=3
&EvaluationPeriods=2
&ComparisonOperator=GreaterThanOrEqualToThreshold
&AlarmName=AddCapacityToProcessQueue
&AlarmActions.member.1= arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scaleout-policy
&Namespace=AWS/SQS
&Dimensions.member.1.Name=QueueName
&Dimensions.member.1.Value=MyQueue
&Period=60
&MetricName=ApproximateNumberOfMessagesVisible
&Version=2010-08-01
&Action=PutMetricAlarm
&AUTHPARAMS
```

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

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

2. Use the CloudWatch action `PutMetricAlarm` to create an alarm to decrease the size of the Auto Scaling group when the number of messages in the queue available for processing (`ApproximateNumberOfMessagesVisible`) decreases to one and remains one or less for a period of one minute. Specify the following parameters:

   - AlarmName = `RemoveCapacityFromTheProcessQueue`

- MetricName = `ApproximateNumberOfMessagesVisible`
- Namespace = `"AWS/SQS"`
- Statistic = `Average`
- Period = `60`
- Threshold = `1`
- ComparisonOperator = `LessThanOrEqualToThreshold`
- Dimensions.member.1 = `"QueueName=MyQueue"`
- EvaluationPeriods = `2`
- AlarmActions.member.1 = arn:aws:autoscaling:us-east-1:803981987763:scalingPolicy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-asg:policyName/my-sqs-scalein-policy

Your request should look like the following example:

```
https://monitoring.us-east-1.amazonaws.com/?Statistic=Average
&Threshold=1
&EvaluationPeriods=2
&ComparisonOperator=LessThanOrEqualToThreshold
&AlarmName=RemoveCapacityFromTheProcessQueue
&AlarmActions.member.1= arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scalein-policy
&Namespace=AWS/SQS
&Dimensions.member.1.Name=QueueName
&Dimensions.member.1.Value=MyQueue
&Period=60
&MetricName=ApproximateNumberOfMessagesVisible
&Version=2010-08-01
&Action=PutMetricAlarm
&AUTHPARAM
```

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

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

Verify Your Scaling Policies and CloudWatch Alarms

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

To verify your CloudWatch alarms

1. Call the CloudWatch action `DescribeAlarms`, and specify the following parameters:

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

   Your request should look like the following example:

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

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

```
<DescribeAlarmsResponse xmlns="http://monitoring.amazonaws.com/doc/2010-08-
01/">
  <DescribeAlarmsResult>
    <MetricAlarms>
      <member>
        <StateUpdatedTimestamp>2013-02-25T21:36:22.428Z</StateUpdated
Timestamp>
        <InsufficientDataActions/>
        <AlarmArn>arn:aws:cloudwatch:us-east-1:803981987763:alarm:AddCapa
cityToProcessQueue</AlarmArn>
        <AlarmConfigurationUpdatedTimestamp>2013-02-
25T21:36:22.428Z</AlarmConfigurationUpdatedTimestamp>
        <AlarmName>AddCapacityToProcessQueue</AlarmName>
        <StateValue>INSUFFICIENT_DATA</StateValue>
        <Period>60</Period>
        <OKActions/>
        <ActionsEnabled>true</ActionsEnabled>
        <Namespace>AWS/SQS</Namespace>
        <Threshold>3.0</Threshold>
        <EvaluationPeriods>2</EvaluationPeriods>
        <Statistic>Average</Statistic>
        <AlarmActions>
          <member>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scaleout-policy</member>
        </AlarmActions>
        <StateReason>Unchecked: Initial alarm creation</StateReason>
        <Dimensions>
          <member>
            <Name>AutoScalingGroupName</Name>
            <Value>my-test-asg</Value>
          </member>
        </Dimensions>
       <ComparisonOperator>GreaterThanOrEqualToThreshold</ComparisonOperator>

        <MetricName>ApproximateNumberOfMessagesVisible</MetricName>
      </member>
      <member>
        <StateUpdatedTimestamp>2013-02-25T23:28:12.092Z</StateUpdated
Timestamp>
        <InsufficientDataActions/>
       <AlarmArn>arn:aws:cloudwatch:us-east-1:803981987763:alarm:RemoveCa
pacityFromProcessQueue</AlarmArn>
        <AlarmConfigurationUpdatedTimestamp>2013-02-
25T23:28:12.092Z</AlarmConfigurationUpdatedTimestamp>
```

```
        <AlarmName>RemoveCapacityFromProcessTheQueue</AlarmName>
        <StateValue>INSUFFICIENT_DATA</StateValue>
        <Period>60</Period>
        <OKActions/>
        <ActionsEnabled>true</ActionsEnabled>
        <Namespace>AWS/SQS</Namespace>
        <Threshold>1.0</Threshold>
        <EvaluationPeriods>2</EvaluationPeriods>
        <Statistic>Average</Statistic>
        <AlarmActions>
          <member>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-sqs-scalein-policy</member>
        </AlarmActions>
        <StateReason>Unchecked: Initial alarm creation</StateReason>
        <Dimensions>
          <member>
            <Name>QueueName</Name>
            <Value>MyQueue</Value>
          </member>
        </Dimensions>
       <ComparisonOperator>LessThanOrEqualToThreshold</ComparisonOperator>

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

To verify your scaling policies

1. Call the Auto Scaling action `DescribePolicies` by specifying the following parameter:

   - AutoScalingGroupName = `my-test-asg`

   Your request should look like the following example:

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

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

   ```
   <DescribePoliciesResponse xmlns="http://autoscaling.amazonaws.com/doc/2011-
   01-01/">
     <DescribePoliciesResult>
       <ScalingPolicies>
   ```

```
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:4590e0ea-77dd-4f23-bef7-5558c3c8cfc1:autoScalingGroupName/my-test-
asg:policyName/my-
sqs-scalein-policy</PolicyARN>
        <AdjustmentType>ChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>1</ScalingAdjustment>
        <PolicyName>my-sqs-scalein-policy</PolicyName>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Alarms>
          <member>
            <AlarmName>RemoveCapacityFromTheProcessQueue</AlarmName>
           <AlarmARN>arn:aws:cloudwatch:us-east-1:803981987763:alarm:Remove
CapacityFromProcessQueue</AlarmARN>
          </member>
        </Alarms>
      </member>
      <member>
        <PolicyARN>arn:aws:autoscaling:us-east-1:803981987763:scaling
Policy:f4390e81-9a48-4655-ba57-f059d17799ea:autoScalingGroupName/my-test-
asg:policyName/my-
sqs-scaleout-policy</PolicyARN>
        <AdjustmentType>ChangeInCapacity</AdjustmentType>
        <ScalingAdjustment>1</ScalingAdjustment>
        <PolicyName>my-sqs-scaleout-policy</PolicyName>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <Alarms>
          <member>
            <AlarmName>AddCapacityToProcessQueue</AlarmName>
           <AlarmARN>arn:aws:cloudwatch:us-east-1:803981987763:alarm:AddCa
pacityToProcessQueue</AlarmARN>
          </member>
        </Alarms>
      </member>
    </ScalingPolicies>
  </DescribePoliciesResult>
  <ResponseMetadata>
    <RequestId>01759ddc-7fa6-11e2-8dbf-5fe8fEXAMPLE</RequestId>
  </ResponseMetadata>
</DescribePoliciesResponse>
```

Test for Scaling Out and Scaling In

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

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

1. Follow the steps in Sending a Message to add messages in the SQS queue you created earlier for this example. Make sure you have at least three messages in the queue.

2. It takes a few minutes for the SQS queue metric `ApproximateNumberOfmessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to launch one EC2 instance.
3. Call the `DescribeAutoScalingGroups` action by specifying the following parameter:

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

   Your request should look similar to the following example:

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

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

```
<DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-21T23:06:56.574Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-2cd22f5c</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
          <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-5a277829</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:bf6a47f1-1eda-4108-bf48-8d8da0c2b72e:auto
ScalingGroupName/my-test-asg</
```

```
AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>10</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>7e20e72f-851c-11e2-b688-ff8f1EXAMPLE</RequestId>
  </ResponseMetadata>
  </DescribeAutoScalingGroupsResponse>
```

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

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

1. Follow the steps in Deleting a Message to remove messages from the SQS queue. Make sure you have no more than one message in the queue.
2. It takes a few minutes for the SQS queue metric `ApproximateNumberOfmessagesVisible` to invoke the CloudWatch alarm. After the CloudWatch alarm is invoked, it notifies the Auto Scaling policy to terminate one EC2 instance.
3. Call the `DescribeAutoScalingGroups` action by specifying the following parameter:

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

   Your request should look similar to the following example:

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

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

```
<DescribeAutoScalingGroupsResult>
    <AutoScalingGroups>
      <member>
        <Tags/>
        <SuspendedProcesses/>
        <AutoScalingGroupName>my-test-asg</AutoScalingGroupName>
        <HealthCheckType>EC2</HealthCheckType>
        <CreatedTime>2013-01-21T23:06:56.574Z</CreatedTime>
        <EnabledMetrics/>
        <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
        <Instances>
         <member>
            <HealthStatus>Healthy</HealthStatus>
            <AvailabilityZone>us-east-1e</AvailabilityZone>
            <InstanceId>i-5a277829</InstanceId>
            <LaunchConfigurationName>my-test-lc</LaunchConfigurationName>
```

```
            <LifecycleState>InService</LifecycleState>
          </member>
        </Instances>
        <DesiredCapacity>1</DesiredCapacity>
        <AvailabilityZones>
          <member>us-east-1e</member>
        </AvailabilityZones>
        <LoadBalancerNames/>
        <MinSize>1</MinSize>
        <VPCZoneIdentifier/>
        <HealthCheckGracePeriod>0</HealthCheckGracePeriod>
        <DefaultCooldown>300</DefaultCooldown>
        <AutoScalingGroupARN>arn:aws:autoscaling:us-east-
1:803981987763:autoScalingGroup:bf6a47f1-1eda-4108-bf48-8d8da0c2b72e:auto
ScalingGroupName/my-test-asg</
AutoScalingGroupARN>
        <TerminationPolicies>
          <member>Default</member>
        </TerminationPolicies>
        <MaxSize>10</MaxSize>
      </member>
    </AutoScalingGroups>
  </DescribeAutoScalingGroupsResult>
  <ResponseMetadata>
    <RequestId>7e20e72f-851c-11e2-b688-ff8f1EXAMPLE</RequestId>
  </ResponseMetadata>
  </DescribeAutoScalingGroupsResponse>
```

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

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

Get Email Notifications When Your Auto Scaling Group Changes

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

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

- An Amazon Resource Name (ARN), which you generate when you create an Amazon SNS topic. An endpoint, such as an email address, must be subscribed to the topic in order for the endpoint to receive messages published to the topic. In this section, we assume that you already have created an Amazon SNS topic and that you have an ARN. For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.
- An Auto Scaling Group, which you created when you went through Basic Auto Scaling Configuration (p. 32).

- A notification configuration. You configure an Auto Scaling group to send notifications when specified events take place by calling the `as-put-notification-configuration` CLI command or the `PutNotificationConfiguration` API action. We discuss the steps for setting up a notification configuration in Set Up the Notification Configuration (p. 158). For more information about the command, go to PutNotificationConfiguration in the *Auto Scaling API Reference*.
- A list of Auto Scaling notification types, which are events that will cause the notification to be sent. The following table lists the available notification types:

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

  For the most updated list of notification types, use the `as-describe-auto-scaling-notification-types` CLI command or the DescribeAutoScalingNotificationTypes API action.

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

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

Tools You Will Use

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

You will be using the following commands.

Command	Description
`as-put-notification-configuration`	Configures an Auto Scaling group to send notifications when specified events take place.
`as-describe-notification-configurations`	Returns a list of notification actions associated with Auto Scaling groups for specified events.
`as-describe-auto-scaling-notification-types`	Returns a list of all notification types that are supported by Auto Scaling.
`as-delete-notification-configuration`	Deletes notifications created by `PutNotificationConfiguration`.

Command	Description
`as-delete-auto-scaling-group`	Deletes the specified Auto Scaling group.
`as-set-desired-capacity`	Sets the desired capacity of the Auto Scaling group.

Set Up the Notification Configuration

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

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

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

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

For this example, specify:

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

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

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

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

Auto Scaling returns the following:

```
OK-Put Notification Configuration
```

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

Verify Notification Configuration

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

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

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

If you specify the Auto Scaling group, this command returns a full list of all notification configurations for the Auto Scaling group listed. If you don't provide an Auto Scaling group name, the service returns the full details of all Auto Scaling groups. The command also returns a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the `next-token` argument. For this example, specify:

- Auto Scaling group name: `MyGroup`

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

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

Auto Scaling returns the following:

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

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

Update and Generate Notifications

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

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

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

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

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

Open a command prompt and enter the commands.

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

Auto Scaling returns the following:

```
OK-Desired Capacity Set
```

Within a few minutes of calling `as-set-desired-capacity`, you should get an email notification that instances for `MyGroup` were launched. When you receive this notification, you can confirm it by using `as-describe-auto-scaling-groups` and specifying `MyGroup` as the Auto Scaling group name.

Delete Notification Configuration

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

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

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

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

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

Open a command prompt and enter the command:

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

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

```
OK-Deleted Notification Configuration
```

Delete Auto Scaling Group

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

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

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

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

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

Open a command prompt and enter the following command:

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

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

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

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

The command looks similar to the following example:

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

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

Tasks Completed

You just performed the following tasks:

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

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

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

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

Launch Spot Instances in Your Auto Scaling Group

If you can be flexible about when you run your applications, or if the performance of your application will scale efficiently with the addition of temporary compute resources, and you want to benefit from the cost-savings of using Amazon EC2 Spot Instances, you can set up Auto Scaling to launch Spot Instances For more information about Spot Instances, go to Spot Instances in the *Amazon Elastic Compute Cloud User Guide*.

Spot Instances are a way for you to to purchase unused Amazon EC2 capacity. You bid on certain instances, and, as long as your bid price exceeds the current Spot price for those instances, you get to use them.

When you use Spot Instances with Auto Scaling, first you need to understand the basics of Spot Instance bids and how they interact with Auto Scaling.

- **Spot market price and your bid price.** If the market price for Spot Instances rises above your Spot bid price for a running instance, Amazon EC2 will terminate your instance. This is true for all Spot

Instances, whether or not you manage them using Auto Scaling. If your Spot bid price exactly matches the Spot market price, your bid may or may not be fulfilled, depending on several factors—such as available Spot Instance capacity.

- **Setting your bid price.** When you use Auto Scaling to launch Spot Instances, you set your Spot bid price in an Auto Scaling launch configuration.
- **Changing your bid price.** If you want to change your Spot bid price, you have to create a new launch configuration and associate it with your Auto Scaling group. You cannot edit launch configurations, which can serve as a record of the configuration history for running and terminated instances and their bid price.
- **New bid price and running instances.** When you change your Spot bid price by creating a new launch configuration, running instances will continue to run as long as the Spot bid price for those running instances is higher than the current Spot market price.
- **Maintaining your Spot Instances.** When your instance is terminated (whether it's a Spot Instance or an On-Demand instance), Auto Scaling will launch another instance in an effort to replace it and maintain the desired capacity specified in the Auto Scaling group. However, whether or not Auto Scaling successfully launches an instance depends on the Spot bid price as compared to the Spot market price: If the bid price is higher than the market price, then an instance will be launched; if the market price is higher than the bid price, then no instance will be launched.
- **Auto Scaling and Spot Instance termination of instances.** Amazon EC2 terminates a Spot Instance when the bid price for that instance falls below the Spot market price. Auto Scaling terminates or replaces instances based on other criteria. For more information, see Auto Scaling Instance Termination.

In this section, we will create a launch configuration and an Auto Scaling group that launches Spot Instances. We will use the Auto Scaling command line interface (CLI) to create the launch configuration and the Auto Scaling group, and to verify and obtain information about the new instances. In this scenario, we will perform the following tasks:

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

Tools You Will Use

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

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

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

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

Create a Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Auto Scaling Configuration (p. 32). Use the basic scenario to get started with the infrastructure that is typically needed in Auto Scaling.

In Auto Scaling, you place bids for Spot Instances using the `as-create-launch-config` command. Specify the maximum price you are willing to pay for an instance using the `--spot-price` option. Auto Scaling will request Spot Instances using this price, but this price is not what you actually pay for the instances when they are launched. You pay the current Spot price as long as it is lower than your bid price. For example, say you bid $0.05 for m1.small instances. Your bid gets fulfilled if the current Spot market price for an m1.small Spot Instance is $0.03, or any other price below $0.05, and you will be charged the current market price of $0.03 per hour. In fact, as long as your Spot bid is higher than the Spot market price, your bid will be fulfilled and a Spot Instance will be launched for you. So, if the current Spot market price drops to $0.02 or rises to $0.04, you will pay the new price.

It is important that you carefully consider the price you specify for your bid. Only specify Spot bid prices that you are willing to pay. If you intentionally bid too high, you might end up paying your high bid price if the Spot market price rises. Also, if you specify a new Spot bid price that is higher than the bid price for Spot Instances that you are currently running, your new Spot bid may result in your own running instances being replaced at the higher price. To help you choose the appropriate price to bid, you can view the Spot price history using the AWS Management Console, CLI, or API. For more information, see View Spot Price History in the *Amazon Elastic Compute Cloud User Guide*.

In addition, if you want to specify a new Spot bid price, you have to create a new launch configuration, and update your Auto Scaling group to specify the new launch configuration. Launch configurations represent a record of the details of running and terminated instances. They can be helpful in tracking the history of your instances. For more information about changing Spot bids, see Update the Bid Price for the Spot Instances (p. 169) later in this walkthrough.

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

- Launch configuration name = `spotlc-5cents`

Note
When Auto Scaling launches instances, it does not distinguish the Spot Instances from the On-Demand instances. To help you identify which of your instances are Spot Instances, refer to their launch configurations. Consider assigning a launch configuration name that includes *spot* and the bid price.

- Image ID = `ami-e565ba8c`
  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- Instance type = `m1.small`
- Spot price = `$0.05`

Your command should look similar to the following example:

```
as-create-launch-config spotlc-5cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.05"
```

You should get a confirmation like the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Create your Auto Scaling group by using `as-create-auto-scaling-group` and then specifying the launch configuration you just created. In addition, let Auto Scaling know at what level it should maintain your fleet of Spot Instances by setting `min-size` and `desired-capacity`. If your Spot bid price falls below the Spot price, and Amazon EC2 terminates your instance, Auto Scaling will submit your bid again and launch an instance in an effort to maintain the desired capacity you specified. However, whether or not Auto Scaling successfully launches an instance depends on the Spot bid price as compared to the Spot market price: If the Spot bid price is higher than the Spot market price, then an instance will be launched; if the market price is higher than the bid price, then no instance will be launched at that point. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 34).

Specify these values for the command:

- Auto Scaling Group name = `spotasg`
- Launch configuration name = `spotlc-5cents`
- Availability Zone = `us-east-1a,us-east-1b`
- Max size = `5`
- Min size = `1`
- Desired capacity = `3`

Your command should look like the following example:

```
as-create-auto-scaling-group spotasg --launch-configuration spotlc-5cents --
availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 1 --desired-
capacity 3
```

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

```
OK-Created AutoScalingGroup
```

Verifying and Checking Your Instances

You might want to verify details of your Auto Scaling group after you've set it up.

- Check whether Auto Scaling is submitting your bids successfully.

  The `as-describe-scaling-activities` command is a useful tool for this task. The command lists the activities that Auto Scaling performed for a specified Auto Scaling group.

  This is the basic syntax:

  as-describe-scaling-activities [*ActivityIds* [*ActivityIds* ...]]
  [--auto-scaling-group *value*] [--max-records *value*] [General Options]

  Specifying the Auto Scaling group and the Activity ID is optional. Activity ID is an identifier for an Auto Scaling activity.

  If you don't specify the Auto Scaling group, the command will return all activities for all Auto Scaling groups associated with your account. If you specify the Auto Scaling group, only the activities for that Auto Scaling group will be listed.

  In this walkthrough, we are using the `as-describe-scaling-activities` command to see the state of your bid. Your command should look like the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --headers
```

  If not all your bids are fulfilled, you will get information that looks like the following example:

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

  In this response, you know that your bids were placed, one of the bids is successful, and Auto Scaling is waiting for the other two bids.

  Note
  If the `as-describe-scaling-activities` command returns a list that includes *Failed* activities, check the data you specified in the launch configuration. For example:

  - The Amazon Machine Image (AMI) might not be valid anymore.
  - The Spot bid price specified in the launch configuration is lower than the Spot market price.

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

```
ACTIVITY  ACTIVITY-ID                              END-TIME             GROUP-
NAME      CODE
ACTIVITY  90630906-b40f-41a6-967a-cd6534b2dfca  2012-06-01T02:32:15Z  spotasg
         Successful
```

```
ACTIVITY  a1139948-ad0c-4600-9efe-9dab8ce23615  2012-06-01T00:48:02Z  spotasg
          Successful
ACTIVITY  33001e70-6659-4494-a817-674d1b7a2f58  2012-06-01T02:31:11Z  spotasg
          Successful
```

The output shows that the listed activities were successful. Because we know that *spotasg* is an Auto Scaling group that uses a launch configuration with a Spot bid price, you can assume that the activities represent the bids placed by Auto Scaling.

- Find out more information about a scaling activity.

  You can get more details about the activities and instances by using the `--show-xml` option of `as-describe-scaling-activities`. Your command should look like the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --show-xml
```

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

```
<DescribeScalingActivitiesResponse xmlns="http://autoscaling.amazon
aws.com/doc/2011-01-01/">
  <DescribeScalingActivitiesResult>
    <NextToken>b5a3b43e-10c6-4b61-8e41-2756db1fb8f5</NextToken>
    <Activities>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>90630906-b40f-41a6-967a-cd6534b2dfca</ActivityId>
        <StartTime>2012-06-01T00:48:21.942Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
       <Cause>At 2012-06-01T00:48:21Z a difference between desired and actual
 capacity changing the desired capacity, increasing the capacity from 2 to
3.</Cause>
        <Details>{}</Details>
        <Description>Launching a new EC2 instance: i-fe30d187</Description>
        <EndTime>2012-06-01T02:32:15Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>a1139948-ad0c-4600-9efe-9dab8ce23615</ActivityId>
        <StartTime>2012-06-01T00:47:51.293Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
        <Cause>At 2012-06-01T00:47:51Z an instance was taken out of service
in response to a system health-check.</Cause>
        <Details>{}</Details>
        <Description>Terminating EC2 instance: i-88ce28f1</Description>
        <EndTime>2012-06-01T00:48:02Z</EndTime>
      </member>
      <member>
        <StatusCode>Successful</StatusCode>
        <Progress>0</Progress>
        <ActivityId>33001e70-6659-4494-a817-674d1b7a2f58</ActivityId>
        <StartTime>2012-06-01T00:46:19.723Z</StartTime>
        <AutoScalingGroupName>spotasg</AutoScalingGroupName>
       <Cause>At 2012-06-01T00:46:19Z a difference between desired and actual
 capacity changing the desired capacity, increasing the capacity from 2 to
3.</Cause>
```

```
        <Details>{}</Details>
        <Description>Launching a new EC2 instance: i-2c30d155</Description>
        <EndTime>2012-06-01T02:31:11Z</EndTime>
      </member>
      ...
    </Activities>
  </DescribeScalingActivitiesResult>
  <ResponseMetadata>
    <RequestId>d02af4bc-ad8f-11e1-85db-83e1968c7d8d</RequestId>
  </ResponseMetadata>
</DescribeScalingActivitiesResponse>
```

The XML output shows more detail about the Spot and Auto Scaling activity.

- `Cause: At 2012-06-01T00:48:21Z a difference between desired and actual capacity changing the desired capacity, increasing the capacity from 2 to 3. Description: Launching a new EC2 instance: i-fe30d187`

  If an instance is terminated and the number of instances falls below the desired capacity, Auto Scaling will launch a new instance so that the total number of your running instances rises back to the level specified for desired capacity.
- `Cause: At 2012-06-01T00:47:51Z an instance was taken out of service in response to a system health-check. Description: Terminating EC2 instance: i-88ce28f1`

  Auto Scaling maintains the desired number of instances by monitoring the health status of the instances in the Auto Scaling group. When Auto Scaling receives notification that an instance is *unhealthy* or terminated, Auto Scaling launches another instance to take the place of the unhealthy instance. For information about how Auto Scaling monitors the health status of instances, go to Maintain a Fixed Number of Running EC2 Instances (p. 43).

  Note
  Auto Scaling provides the cause of instance termination that is not the result of a scaling activity. This includes instances that have been terminated because the Spot market price exceeded their bid price.

  When Auto Scaling attempts to replace terminated instances resulting from the Spot market price rising above the instances' bid price, Auto Scaling will place the bid specified in the current launch configuration and attempt to launch another instance to maintain the desired capacity.

- Confirm that Auto Scaling is launching your Spot Instances according to your specifications.

  To confirm that Auto Scaling is launching the instances you want in the Availability Zones you specified, use `as-describe-auto-scaling-groups`. The command will show details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see Verify Auto Scaling Group Creation (p. 35).

  This is the basic syntax:

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

  Specify the `--headers` general option to show the column headers, which can make the information easier to read.

  Your command should look like the following example:

```
as-describe-auto-scaling-groups spotasg --headers
```

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

```
AUTO-SCALING-GROUP  GROUP-NAME     LAUNCH-CONFIG  AVAILABILITY-ZONES    MIN-
SIZE  MAX-SIZE  DESIRED-CAPACITY
AUTO-SCALING-GROUP  spotasg         spotlc-5cents  us-east-1b,us-east-1a  1
        5         3
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-CONFIG
INSTANCE  i-2c30d155   us-east-1a         InService  Healthy  spotlc-5cents
INSTANCE  i-fe30d187   us-east-1a         InService  Healthy  spotlc-5cents
INSTANCE  i-c630d1bf   us-east-1a         InService  Healthy  spotlc-5cents
```

You can see that Auto Scaling launched 3 instances in us-east-1a, as you specified, and they are all running.

- Get details about your Spot Instances.

  In addition to using `as-describe-auto-scaling-groups`, you can find out details about the Spot Instances launched for you by Auto Scaling, by using the `as-describe-auto-scaling-instances` command.

  This is the basic syntax:

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

  Specifying `InstanceIds` is optional. If you specify it, the command will return information about the instance, if it exists. If you don't specify `InstanceIds`, the command returns information about all instances associated with your Auto Scaling account.

  In this walkthrough, we are assuming that you created one launch configuration and Auto Scaling group, and you want to find out details about all your Spot Instances.

  Your command should look like the following example:

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

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

```
INSTANCE  INSTANCE-ID  GROUP-NAME     AVAILABILITY-ZONE  STATE      STATUS
  LAUNCH-CONFIG
INSTANCE  i-2c30d155   spotasg        us-east-1a         InService  HEALTHY
  spotlc-5cents
INSTANCE  i-c630d1bf   spotasg        us-east-1a         InService  HEALTHY
  spotlc-5cents
INSTANCE  i-fe30d187   spotasg        us-east-1a         InService  HEALTHY
  spotlc-5cents
```

Get Notifications When the Auto Scaling Group Changes

Optionally, you may want to receive notifications when instances are terminated and launched.

When the Spot market price rises above the Spot bid price of your running instances, Amazon EC2 terminates these instances. If your Spot Instances are terminated, Auto Scaling will try to launch

replacement instances and satisfy the desired capacity you specified for your Auto Scaling group. You can set up Auto Scaling to notify you using Amazon Simple Notification Service (Amazon SNS) when instance launch or termination events occur.

To do this, you will need the following:

- An Amazon Resource Name (ARN), which you generate when you create an Amazon Simple Notification Service (Amazon SNS) topic. An endpoint, such as an email address, must be subscribed to the topic in order for the endpoint to receive messages published to the topic. For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.
- An Auto Scaling Group, which you created earlier in this walkthrough.
- A notification configuration. You configure an Auto Scaling group to send notifications when specified events take place by calling the `as-put-notification-configuration` CLI command or the `PutNotificationConfiguration` API action. For more information about the command, go to PutNotificationConfiguration in the *Auto Scaling API Reference*.
- A list of Auto Scaling notification types, which are events that will cause the notification to be sent.

For information about setting up email notifications in Auto Scaling, go to Get Email Notifications When Your Auto Scaling Group Changes (p. 156).

Update the Bid Price for the Spot Instances

Auto Scaling launch configurations cannot be changed. They represent a record of the launch configuration details of running and terminated instances. They can be helpful in tracking the history of your instances. If you want to modify your bid price for Spot Instances, you must create a new launch configuration.

If, for example, you want to launch a set of Spot Instances that have a higher likelihood of running uninterrupted a long time, you can specify a higher Spot bid price. To do this, you must create a new launch configuration and associate it with your Auto Scaling group, using the same procedure that you followed earlier in this walkthrough.

To update the bid price for Spot Instances

1. Create a new launch configuration specifying the new Spot bid price, by using the `as-create-launch-config` command.

   Specify the following values:

   - Launch configuration name = spotlc-7cents
   - Image ID = ami-e565ba8c
   - Instance type = m1.small
   - Spot price = $0.07

   Your command should look similar to the following example:

   ```
   as-create-launch-config spotlc-7cents --image-id ami-e565ba8c --instance-
   type m1.small --spot-price "0.07"
   ```

2. Modify your Auto Scaling group to specify the new launch configuration, by using the `as-update-auto-scaling-group` command.

   This is the basic syntax:

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

In this walkthrough, the only change to the *spotasg* Auto Scaling group is the launch configuration it uses.

Your command, specifying the *spotasg* Auto Scaling group and the *spotlc-7cents* launch configuration, should look similar to the following example:

```
as-update-auto-scaling-group spotasg --launch-configuration spotlc-7cents
```

3. View your changes.

   You can view the status of your Spot bid and a list of the bids that Auto Scaling placed for you by running `as-describe-scaling-activities` soon after you create your Auto Scaling group.

   Your command should look similar to the following example:

```
as-describe-scaling-activities --auto-scaling-group spotasg --headers
```

   If not all your bids are fulfilled, you will get information that looks like the following example:

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

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

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

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

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

```
          Successful

ACTIVITY  bfb548ad-8bc7-4a78-a7db-3b41f73501fc  2012-05-23T15:38:07Z  spotasg
          Successful

ACTIVITY  82d2b9bb-3d64-46d9-99b6-054a9ecf5ac2  2012-05-23T15:30:28Z  spotasg
          Successful

ACTIVITY  95b7589b-f8ac-49bc-8c83-514bf664b4ee  2012-05-23T15:30:28Z  spotasg
          Successful

ACTIVITY  57bbf77a-99d6-4d94-a6db-76b2307fb9de  2012-05-23T15:16:34Z  spotasg
          Successful
```

Bids are represented by values such as `sir-93828812` and `sir-016cf812`.

When you create a new launch configuration that sets a new bid price for Spot Instances, and you have Spot Instances already running based on a different bid price, these instances based on a different bid price will continue running and will only be terminated if the Spot market price goes above the bid price on which the running Spot Instance was based.

Set Up a Schedule for Your Spot Bids

You can set up Auto Scaling to launch a certain number of instances at a specific time. This capability is useful when you want to take advantage of a window of time when prices are lower, for example, or you want to terminate Spot Instances at a specific time. To set up a schedule, you use `as-put-scheduled-update-group-action`. This is the basic syntax:

as-put-scheduled-update-group-action *ScheduledActionName* --auto-scaling-group *value* [--desired-capacity *value*] [--end-time *value*][--max-size *value*][--min-size *value*] [--recurrence *value*][--start-time *value*][--time *value*][General Options]

In this walkthrough, use the following values to tell Auto Scaling to increase your fleet of instances to 20 within a five-minute period:

- Scheduled action name: `as-spotbid-schedule`
- Auto Scaling group: `spotasg`
- Start time: `2012-05-15T19:10:00Z`
- End time: `2012-05-15T19:15:00Z`
- Desired capacity:`20`

Your command should look similar to the following example:

```
as-put-scheduled-update-group-action as-spotbid-schedule --auto-scaling-group
spotasg --desired-capacity 20 --start-time 2012-05-15T19:10:00Z --end-time 2012-
05-15T19:15:00Z
```

You should get a confirmation like the following example:

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

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

You will get information similar to the following example:

```
UPDATE-GROUP-ACTION spotasg as-spotbid-schedule 2012-05-15T19:10:00Z 20
```

Clean up

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

Run the command with the following values:

- Auto Scaling group name = `spotasg`

  Note
  If you have more than one Auto Scaling group, you must run the command for each group that you want to delete.

- Force delete (optional parameter) = `--force-delete`

Your commands should look like the following example:

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

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

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

Tasks Completed

You just performed the following tasks:

- Created a launch configuration that launched Spot Instances.
- Created an Auto Scaling group.
- Obtained information about your Auto Scaling group and instances.
- Set up notifications.
- Updated the bid price.
- Scheduled spot requests.
- Cleaned up.

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

Note
The instance associated with the Auto Scaling group you just created is not launched immediately. So, if you run the snippet as a single code block, it could take a few minutes before you see the instance information.

```
as-create-launch-config spotlc-5cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.05"
as-create-auto-scaling-group spotasg --launch-configuration spotlc-5cents --
availability-zones "us-east-1a,us-east-1b" --max-size 5 --min-size 1 --desired-
capacity 3
as-describe-scaling-activities --auto-scaling-group spotasg --headers
as-describe-scaling-activities --auto-scaling-group spotasg --show-xml
as-describe-auto-scaling-groups spotasg --headers
as-describe-auto-scaling-instances --headers
as-put-notification-configuration spotasg --topic-arn arn:placeholder:MyTopic
--notification-types autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_IN
STANCE_TERMINATE
as-describe-notification-configurations spotasg -headers
as-create-launch-config spotlc-7cents --image-id ami-e565ba8c --instance-type
m1.small --spot-price "0.07"
as-update-auto-scaling-group spotasg --launch-configuration spotlc-7cents
as-describe-scaling-activities --auto-scaling-group spotasg --headers
as-put-scheduled-update-group-action as-spotbid-schedule --auto-scaling-group
spotasg --desired-capacity 20 --start-time 2012-05-15T19:10:00Z --end-time 2012-
05-15T19:15:00Z
as-delete-auto-scaling-group spotasg --force-delete
```

Control Access to Auto Scaling Resources

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

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

For example, you can use IAM with Auto Scaling to control which users in your AWS Account can create launch configurations, 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

Use 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. 175).

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`–To check for date/time conditions.
- `aws:EpochTime`–To check for date/time conditions using a date in epoch or UNIX time.
- `aws:principaltype`–To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`–To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.
- `aws:SourceArn`–To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see Amazon Resource Name (ARN) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`–To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.
- `aws:UserAgent`–To check the client application that made the request.

- `aws:userid`–To check the user ID of the requester.
- `aws:username`–To check the user name of the requester, if available.

Note
Key names are case sensitive.

Example 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":"*"
      }
   ]
}
```

Launch Auto Scaling Instances with IAM Role

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

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

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

For more information about IAM roles and instance profiles, go to Delegating API Access by Using Roles in *Using IAM*.

Prerequisites: Using IAM

In this section, we walk you through the steps for launching Auto Scaling instances with an IAM role. Before you walk, be sure you've completed the following steps using IAM.

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

For information and instructions on creating and managing an IAM role using the IAM console, the IAM CLI, or the IAM Query API, go to Create a Role in *Using IAM*.

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

Steps for Launching Auto Scaling Instances with An IAM role

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

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

Commands You Will Use

You will use the following Auto Scaling commands.

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

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

Create Launch Configuration

If you're not familiar with how to create a launch configuration or an Auto Scaling group, we recommend that you go through the steps in the Basic Auto Scaling Configuration (p. 32). Use the basic scenario to get started with the infrastructure that you need in most Auto Scaling scenarios.

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

- Launch configuration name = `lc-with-instance-profile`
- Image ID = `ami-baba68d3`
  If you don't have an AMI, and you want to find a suitable one, see Amazon Machine Images (AMIs).
- Instance type = `m1.small`
- Instance profile name = `mytest-instance-profile`.

Your command should look similar to the following example:

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

You should get a confirmation like the following example:

```
OK-Created launch config
```

Create an Auto Scaling Group

Create your Auto Scaling group by using `as-create-auto-scaling-group` and then specifying the launch configuration you just created. For more detailed information on the syntax of the `as-create-auto-scaling-group` command, go to Create an Auto Scaling Group (p. 34).

Specify these values for the command:

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

Your command should look like the following example:

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

You should get confirmation like the following example:

```
OK-Created AutoScalingGroup
```

Verify that EC2 Instance Launches with IAM Role

To confirm that Auto Scaling launches your EC2 instances using the IAM role you specify, use `as-describe-auto-scaling-groups`. The command shows details about the group and instances launched. For information about the `as-describe-auto-scaling-groups` command, see Verify Auto Scaling Group Creation (p. 35).

Your command should look like the following example.

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

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

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

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

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

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

Clean Up

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

Run the command with the following values:

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

Your command should look like the following example:

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

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

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

Monitor Your 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. 181).

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. 181).
3. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will disable detailed monitoring for new instances that it creates.
4. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

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

5. If you have Amazon CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the Amazon CloudWatch API to update each alarm so that the alarm's period value is set to 300 seconds.

   Important
   If you do not update your alarms to match the five-minute data aggregations, your alarms will continue to check for statistics every minute and might find no data available for as many as four out of every five periods.

For more information on instance metrics for Amazon EC2 instances, go to Amazon CloudWatch Developer Guide.

Auto Scaling Group Metrics

Group metrics are metrics that Auto Scaling group sends to Amazon CloudWatch to describe the group rather than any of its instances. If you enable group metrics, Auto Scaling sends aggregated data to Amazon CloudWatch every minute. If you disable group metrics, Auto Scaling does not send any group metrics data to Amazon CloudWatch.

To enable group metrics

1. Enable detailed instance monitoring for the Auto Scaling group by setting the *`InstanceMonitoring.Enabled`* flag in the Auto Scaling group's launch configuration. For more information, see Monitor Your Auto Scaling Instances (p. 179).
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.
`GroupMaxSize`	The maximum size of the Auto Scaling group.
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain.
`GroupInServiceInstances`	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating.
`GroupPendingInstances`	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating.
`GroupTerminatingInstances`	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.
`GroupTotalInstances`	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating.

Dimensions for Auto Scaling Group Metrics

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

Troubleshooting Auto Scaling

Amazon Web Services provides specific and descriptive errors to help you troubleshoot Auto Scaling problems. The error messages can be retrieved from the description of the Auto Scaling activities. You can use either the Query API or the command line interface (CLI) to retrieve an error message.

Retrieving an Error Message

To retrieve an error message from the description of Auto Scaling activities, use the `as-describe-scaling-activities` command. Alternatively, you can use the `DescribeScalingActivities` API action. For more information about the API action, go to DescribeScalingActivities in the *Auto Scaling API Reference*.

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

```
as-describe-scaling-activities [ActivityIds [ ActivityIds...] ]
[--auto-scaling-group value][--max-records value][General Options]
```

In this example, you'll get the xml description of the Auto Scaling activities for the `MyASGroup` Auto Scaling group.

```
PROMPT>as-describe-scaling-activities  --auto-scaling-group MyASGroup --show-
xml
```

Auto Scaling returns the following:

```
<DescribeScalingActivitiesResponse xmlns="http://ec2.amazonaws.com/doc/2011-01-
01/">
<DescribeScalingActivitiesResult>
<Activities>
   <member>
     <StatusCode>Failed</StatusCode>
     <Progress>0</Progress>
     <ActivityId>063308ae-aa22-4a9b-94f4-9fae70b82ad0</ActivityId>
     <StartTime>2012-04-12T17:32:07.882Z</StartTime>
     <AutoScalingGroupName>MyASGroup</AutoScalingGroupName>
     <Cause>At 2012-04-12T17:31:30Z a user request created an AutoScalingGroup
 changing the desired capacity from 0 to 1.  At 2012-04-12T17:32:07Z an instance
```

```
 was started in response to a difference between desired and actual capacity,
increasing the capacity from 0 to 1.</Cause>
     <Details>{}</Details>
     <Description>Launching a new EC2 instance.  Status Reason: The image id
'ami-4edb0327' does not exist. Launching EC2 instance failed.</Description>
     <EndTime>2012-04-12T17:32:08Z</EndTime>
     <StatusMessage>The image id 'ami-4edb0327' does not exist. Launching EC2
instance failed.</StatusMessage>
   </member>
</Activities>
  </DescribeScalingActivitiesResult>
  <ResponseMetadata>
   <RequestId>7a641adc-84c5-11e1-a8a5-217eb05262e2</RequestId>
  </ResponseMetadata>
</DescribeScalingActivitiesResponse>
```

The response includes a list of `Activities` associated with the `MyASGroup` Auto Scaling group. The `StatusCode` contains the current status of the activity. The `StatusMessage` contains the error message, which is the verbose description of the activity status.

Troubleshooting Auto Scaling issues involves looking at how your Amazon EC2 AMIs and instances are configured. You can create, access, and manage your AMIs and instances using one of the Amazon EC2 interfaces: the AWS Management Console, the command line interface (CLI), or the Query API. You will have to install the Amazon EC2 command line tools before you can use them. For more information go to Getting Started with the Command Line Tools in the *Amazon Elastic Compute Cloud User Guide*. For information on using the Query API, go to Making API Requests in the *Amazon Elastic Compute Cloud User Guide*.

The following tables list the types of error messages and provide links to the troubleshooting resources that you can use as you work with your Auto Scaling issues.

Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure

Issues with	Error Message
Auto Scaling group	AutoScalingGroup <Auto Scaling group name> not found. (p. 187)
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request ..... (p. 187)
AWS account	You are not subscribed to this service. Please go to http://aws.amazon.com. (p. 187)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 187)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 188)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 188)
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>).... (p. 187)
Key pair	The key pair <key pair associated with your Amazon EC2 instance> does not exist. Launching EC2 instance failed. (p. 186)
Launch configuration	The requested configuration is currently not supported. (p. 186)

Issues with	Error Message
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 188)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 186)

Troubleshooting Auto Scaling: Amazon EC2 AMIs

Issues with	Error Message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 189)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 189)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 190)
Architecture mismatch	The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching ec2 instance failed. (p. 190)
Virtualization type	Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed. (p. 189)

Troubleshooting Auto Scaling: Load Balancer Configuration

Issues with	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 190)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 191)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 191)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 191)

Troubleshooting Auto Scaling: Capacity Limits

Issues with	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 192)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>).... (p. 192)

Troubleshooting Auto Scaling: Amazon EC2 Instance Launch Failure

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

When your Amazon EC2 instances fail to launch, you might get one or more of the error messages covered in the following topics. To retrieve an error message and to review the error message lists sorted by the type of issue, see Retrieving an Error Message (p. 183).

The security group <name of the security group> does not exist. Launching EC2 instance failed.

- **Cause**: The security group specified in your launch configuration might have been deleted.
- **Solution**:
  1. Use the DescribeSecurityGroups action or ec2-describe-group command to get the list of the security groups associated with your account.
  2. From the list, select the security groups you want to use. To create a new security group use the CreateSecurityGroup action or the ec2-create-group command.
  3. Create a new launch configuration.
  4. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

The key pair <key pair associated with your Amazon EC2 instance> does not exist. Launching EC2 instance failed.

- **Cause**: The key pair that was used when launching the instance might have been deleted.
- **Solution**:
  1. Use the DescribeKeyPairs action or the ec2-describe-kepairs command to get the list of the key pairs available to you.
  2. From the list, select the key pairs you want to use. To create a new key pair use CreateKeyPair action or ec2-create-keypair command.
  3. Create a new launch configuration.
  4. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

The requested configuration is currently not supported.

- **Cause**: Some fields in your launch configuration might not be currently supported.
- **Solution**:
  1. Create a new launch configuration.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

AutoScalingGroup <Auto Scaling group name> not found.

- **Cause**: The Auto Scaling group might have been deleted.
- **Solution**: Create a new Auto Scaling group.

The requested Availability Zone is no longer supported. Please retry your request .....

- **Error Message**: The requested Availability Zone is no longer supported. Please retry your request by not specifying an Availability Zone or choosing <list of available Availability Zones>. Launching EC2 instance failed.
- **Cause**: The Availability Zone associated with your Auto scaling group might not be currently available.
- **Solution**: Update your Auto Scaling group with the recommendations in the error message.

Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)....

- **Error Message**: Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>). Please retry your request by not specifying an Availability Zone or choosing <list of Availability Zones that supports the instance type>. Launching EC2 instance failed.
- **Cause**: The instance type associated with your launch configuration might not be currently available in the Availability Zones specified in your Auto Scaling group.
- **Solution**: Update your Auto Scaling group with the recommendations in the error message.

You are not subscribed to this service. Please go to http://aws.amazon.com.

- **Cause**: Your AWS account might have expired.
- **Solution**:Go to http://aws.amazon.com and click the **Sign Up Now** button to open a new account.

Invalid device name upload. Launching EC2 instance failed.

- **Cause**: The block device mappings in your launch configuration might contain block device names that are not available or currently not supported.
- **Solution**:
  1. Use the AWS Management Console, the DescribeVolumes action, or the ec2-describe-volumes command to see how the volumes are exposed to the instance.
  2. Create a new launch configuration using the device name listed in the volume description.

3. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

Value (<name associated with the instance storage device>) for parameter virtualName is invalid...

- **Error Message**: Value (<name associated with the instance storage device>) for parameter virtualName is invalid. Expected format: 'ephemeralNUMBER'. Launching EC2 instance failed.
- **Cause**: The format specified for the virtual name associated with the block device is incorrect.
- **Solution**:
  1. Create a new launch configuration by specifying the value of the `virtualName` parameter in the format: `ephemeral<number>`. For information on the device name format, go to Instance Store Device Names in the *Amazon Elastic Compute Cloud User Guide*.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

EBS block device mappings not supported for instance-store AMIs.

- **Cause**: The block device mappings specified in the launch configuration are not supported on your instance.
- **Solution**:
  1. Create a new launch configuration with block device mappings supported by your instance type. For more information on block device mapping, see Block Device Mapping.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.

- **Cause**: Your cluster placement group contains an invalid instance type.
- **Solution**:
  1. For information on valid instance types supported by the the cluster placement groups, go to Using Cluster Instances in the *Amazon Elastic Compute Cloud User Guide*.
  2. Follow instructions detailed in the Creating a Cluster Placement Group section of the *Using Cluster Instances* topic to create a new placement group.
  3. Alternatively, create a new launch configuration with the supported instance type.
  4. Update your Auto Scaling group with new placement group or the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

Troubleshooting Auto Scaling: Amazon EC2 AMIs

The following topics provide information about the issues associated with your Amazon EC2 images, potential causes, and the steps you can take to resolve the issues.

When your Amazon EC2 instances fail to launch due to issues with your AMIs, you might get one or more of the error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see Retrieving an Error Message (p. 183).

The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.

- **Cause**: The AMI might have been deleted after creating the launch configuration.
- **Solution**:
  1. Create a new launch configuration using a valid AMI.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.

- **Cause**: You might have just created your AMI (by taking a snapshot of a running instance or any other way), and it might not be available yet.
- **Solution**: You must wait for your AMI to be available and then create your launch configuration.

Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed.

- **Cause**: The Linux/UNIX AMI with hvm virtualization cannot be used to launch a non-cluster compute instance.
- **Solution**:
  1. Create a new launch configuration using an AMI with a virtualization type of paravirtual to launch a non-cluster compute instance.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

Value (<ami ID>) for parameter virtualName is invalid.

- **Cause**: Incorrect value. The `virtualName` parameter refers to the virtual name associated with the device.
- **Solution**:
  1. Create a new launch configuration by specifying the name of the virtual device of your instance for the `virtualName` parameter.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching ec2 instance failed.

- **Cause**: The architecture of the `InstanceType` mentioned in your launch configuration does not match the image architecture.
- **Solution**:
  1. Create a new launch configuration using the AMI architecture that matches the architecture of the requested instance type.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

Troubleshooting Auto Scaling: Load Balancer Configuration

This following topics provide information about issues caused by the load balancer associated with your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

When your Amazon EC2 instances fail to launch due to issues with the load balancer associated with your Auto Scaling group, you might get one or more of the error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see Retrieving an Error Message (p. 183).

Before you begin troubleshooting issues with the load balancer configurations, be sure you've installed the Elastic Load Balancing interface you plan to use to access your load balancer. For more information, go to Get Set Up With Elastic Load Balancing Interfaces in the *Elastic Load Balancing Developer Guide*.

Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.

- **Cause 1**: The load balancer has been deleted.

- **Solution 1**:
  1. Check to see if your load balancer still exists. You can use either the DescribeLoadBalancer action or the `elb-describe-lbs` command.
  2. If you see your load balancer listed in the response, see **Cause 2**.
  3. If you do not see your load balancer listed in the response, you can either create a new load balancer and then create a new Auto Scaling group or you can create a new Auto Scaling group without the load balancer.
- **Cause 2**: The load balancer name was not specified in the right order when creating the Auto Scaling group.
- **Solution 2**: Create a new Auto Scaling group and specify the load balancer name at the end.

There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.

- **Cause**: The specified load balancer might have been deleted.
- **Solution**: You can either create a new load balancer and then create a new Auto Scaling group or create a new Auto Scaling group without the load balancer.

EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.

- **Cause**: The specified instance does not exist in Amazon VPC.
- **Solution**: You can either delete your load balancer associated with the instance or create a new Auto Scaling group.

The security token included in the request is invalid. Validating load balancer configuration failed.

- **Cause**: Your AWS account might have expired.
- **Solution**: Check if your AWS account is valid. Go to http://aws.amazon.com and click the **Sign Up Now** button to open a new account.

Troubleshooting Auto Scaling: Capacity Limits

The following topics provide information about issues with the capacity limits of your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

When your Amazon EC2 instances fail to launch due to issues with the capacity limits of your Auto Scaling group, you might get one or more of the error messages covered in the following topics. To retrieve the error message and review the error message lists sorted by the type of issue, see Retrieving an Error Message (p. 183).

We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)....

- **Error Message**: We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>). Our system will be working on provisioning additional capacity. You can currently get <instance type> capacity by not specifying an Availability Zone in your request or choosing <list of Availability Zones that currently supports the instance type>. Launching EC2 instance failed.
- **Cause**: At this time, Auto Scaling cannot support your instance type in your requested Availability Zone.
- **Solution**:
  1. Create a new launch configuration by following the recommendations in the error message.
  2. Update your Auto Scaling group with the new launch configuration using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.

<number of instances> instance(s) are already running. Launching EC2 instance failed.

- **Cause**: The Auto Scaling group has reached the limit set by the `DesiredCapacity` parameter.
- **Solution**:
  - Update your Auto Scaling group by providing a new value for the `DesiredCapacity` parameter using the UpdateAutoScalingGroup action or the `as-update-auto-scaling-group` command.
  - If you've reached the limit for number of EC2 instances, see Contact Us and place a request to raise your Amazon EC2 instance limit.

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. 4).
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. 7).
Secret Access Key	Amazon Web Services (AWS) key used for request authentication.
statistic	Amazon CloudWatch key term. A *statistic* is a time-series of values for a metric. For more information, go to the *Amazon CloudWatch Developer Guide*.
trigger	Auto Scaling key term. The mechanism used to initiate Auto Scaling activities.
unbounded	Term used in Web Service Definition Language (WSDL), i.e., `maxOccurs="unbounded"`, meaning that the number of potential occurrences is not limited by a set number. Often used when defining a data type that is a list of other types, such as an unbounded list of integers (element members) or an unbounded list of other complex types that are element/members of the list being defined.
unit	Amazon CloudWatch key term. Every measure that is received by the CloudWatch Service has a unit attached, such as seconds or bytes. For more information,go to the *Amazon CloudWatch Developer Guide*.

Document History

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

- API version: 2011-01-01
- Latest documentation update: June 11, 2013

Change	Description	Release Date
Content restructure	Created a new section for managing Auto Scaling groups and moved all the related procedures in this new section. For more information, see Managing Your Auto Scaling Groups (p. 42).	11 June 2013
New content	Added a walkthrough for using Amazon SQS queues to establish thresholds that Auto Scaling can use to increase or decrease the capacity of your Auto Scaling group. For more information, see Use Amazon SQS Queues to Determine When to Auto Scale (p. 140).	12 March 2013
Content restructure	Changed the title of *Configuring Auto Scaling* to *Configure a Scaling Plan*. Added information about configuring a scaling plan for your Auto Scaling group. For more information, see Configure a Scaling Plan (p. 43).	08 February 2013
Configuring Your Instance Termination Policy	You can now specify the instance termination policy for Auto Scaling to use when terminating Amazon EC2 instances in your Auto Scaling group. For information on configuring your instance termination policy for your Auto scaling group, see Configure Instance Termination Policy for Your Auto Scaling Group (p. 75).	17 September 2012

Change	Description	Release Date
Launching EC2 Instances with an IAM role	You can now use Auto Scaling groups to launch EC2 instances with an IAM instance profile. You can use this feature to assign IAM roles to your instances, allowing applications that run on it to access other AWS services securely. For instructions on launching EC2 instances with an IAM instance profile, see Launch Auto Scaling Instances with IAM Role (p. 176).	11 June 2012
Running Spot Instances	You can now run Spot Instances in Auto Scaling groups by specifying a Spot Instance bid price in your launch configuration. For instructions on launching Spot Instances using Auto Scaling groups, see Launch Spot Instances in Your Auto Scaling Group (p. 161).	7 June 2012
Updated Troubleshooting Auto Scaling section	Added information about the issues, potential causes, and the steps you can take to resolve problems with Auto Scaling. For more information, see Troubleshooting Auto Scaling (p. 183).	15 May 2012
Tagging Auto Scaling Groups and Amazon EC2 Instances	You can now tag Auto Scaling groups and specify that the tag also applies to Amazon EC2 instances that the Auto Scaling group launches after the tag is created. For more information, see Tag Your Auto Scaling Groups and Amazon EC2 Instances (p. 93).	26 January 2012
SNS Integration and Other Features	Auto Scaling now supports Amazon Simple Notification Services (Amazon SNS) so that you can use it to receive notifications whenever Auto Scaling launches or terminates Amazon EC2 instances. For more information, see Get Email Notifications When Your Auto Scaling Group Changes (p. 156). Auto Scaling also added the following new features: • The ability to set up recurring scaling activities using cron syntax. For more information, see the PutScheduledUpdateGroupAction API command. • A new configuration setting that allows you to scale up without adding the launched instance to the load balancer (LoadBalancer). For more information, see the ProcessType API datatype. • The `ForceDelete` flag in the `DeleteAutoScalingGroup` command that tells the service to delete the Auto Scaling Group with the instances associated to it without waiting for the instances to be terminated first. For more information, see the DeleteAutoScalingGroup API command.	20 July 2011
New link	This service's endpoint information is now located in the Amazon Web Services General Reference. For more information, go to Regions and Endpoints in Amazon Web Services General Reference.	2 March 2011

Change	Description	Release Date
Updated example	Added a missing *Dimensions* parameter to examples that call the PutMetricAlarm command. For more information, see Load Balance Your Auto Scaling Group (p. 126).	12 January 2011
New feature	Added the ability to create scheduled scaling actions. For more information, see Working With Auto Scaling (p. 125).	2 December 2010
New feature	Added support for Amazon Virtual Private Cloud (VPC). For more information, see Launch Auto Scaling Instances into Amazon VPC (p. 100).	2 December 2010
New feature	Added support for high performance computing (HPC) clusters.	2 December 2010
New feature	Added the capability to use Elastic Load Balancing Health Check as the health check for Auto Scaling-managed Amazon EC2 instances. For more information, see Add an Elastic Load Balancing Health Check to your Auto Scaling Group (p. 133).	2 December 2010
New design	Removed the older Trigger mechanism and redesigned Auto Scaling to use the CloudWatch alarm feature. For more information, see Working With Auto Scaling (p. 125).	2 December 2010
New feature	Added a new feature that lets you suspend and resume scaling processes. For more information, see Suspendable Processes (p. 9).	2 December 2010
New feature	This service now integrates with AWS Identity and Access Management (IAM). For more information, see Control Access to Auto Scaling Resources (p. 173).	2 December 2010
New Region	Auto Scaling now supports the Asia Pacific (Singapore) Region. For more information, see Regions and Endpoints.	28 April 2010

Index