Auto Scaling

User Guide

Auto Scaling: User Guide

Table of Contents

What Is Auto Scaling?

Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups*. You can specify the minimum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of 1 instance, a desired capacity of 2 instances, and a maximum size of 4 instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.
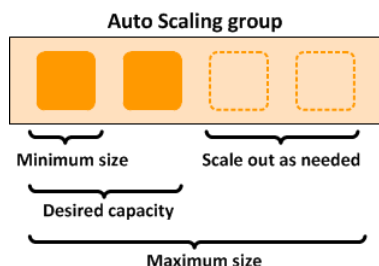
For more information about the benefits of Auto Scaling, see Benefits of Auto Scaling (p. 3).

Auto Scaling Components

The following table describes the key components of Auto Scaling.

	Groups
	Your EC2 instances are organized into *groups* so that they can be treated as a logical unit for the purposes of scaling and management. When you create a group, you can specify its minimum, maximum, and, desired number of EC2 instances. For more information, see Auto Scaling Groups (p. 38).

	Launch configurations Your group uses a *launch configuration* as a template for its EC2 instances. When you create a launch configuration, you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances. For more information, see Launch Configurations (p. 21).
	Scaling plans A *scaling plan* tells Auto Scaling when and how to scale. For example, you can base a scaling plan on the occurrence of specified conditions (dynamic scaling) or on a schedule. For more information, see Scaling Plans (p. 57).

Getting Started

If you're new to Auto Scaling, we recommend that you review Auto Scaling Lifecycle (p. 7) before you begin.

To begin, complete the Getting Started with Auto Scaling (p. 12) tutorial to create an Auto Scaling group and see how it responds when an instance in that group terminates. If you already have running EC2 instances, you can create an Auto Scaling group using an existing EC2 instance, and remove the instance from the group at any time.

Accessing Auto Scaling

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

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

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

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

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

If you prefer to build applications using language-specific APIs instead of submitting a request over HTTP or HTTPS, AWS provides libraries, sample code, tutorials, and other resources for software

developers. These libraries provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses, making it is easier for you to get started. For more information, see AWS SDKs and Tools.

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

Pricing for Auto Scaling

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

PCI DSS Compliance

Auto Scaling supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

Related Services

To automatically distribute incoming application traffic across multiple instances in your Auto Scaling group, use Elastic Load Balancing. For more information, see Elastic Load Balancing User Guide.

To monitor basic statistics for your instances and Amazon EBS volumes, use Amazon CloudWatch. For more information, see the Amazon CloudWatch User Guide.

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

Benefits of Auto Scaling

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

- Better fault tolerance. Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Auto Scaling can launch instances in another one to compensate.
- Better availability. Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demands.
- Better cost management. Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are actually needed and terminating them when they aren't needed.

Contents

Example: Covering Variable Demand

To demonstrate some of the benefits of Auto Scaling, consider a basic Web application running on AWS. This application allows employees to search for conference rooms that they might want to use for meetings. During the beginning and end of the week, usage of this application is minimal. During the middle of the week, more employees are scheduling meetings, so the demands on the application increases significantly.

The following graph shows how much of the application's capacity is used over the course of a week.
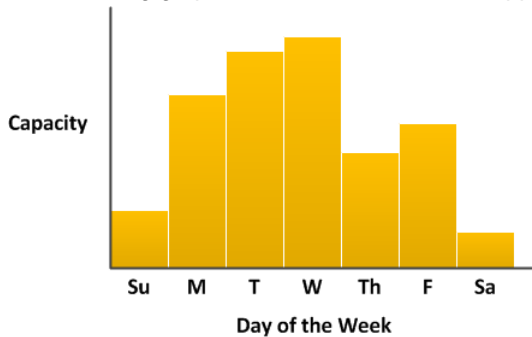
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity. The extra capacity remains unused and, in essence, raises the cost of keeping the application running.
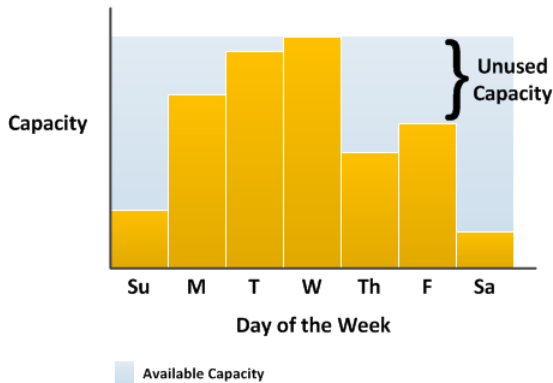
The second option is to have enough capacity to handle the average demands on the application. This option is less expensive, because you aren't purchasing equipment that you'll only use occasionally. However, you risk creating a poor customer experience when the demands on the application exceeds its capacity.
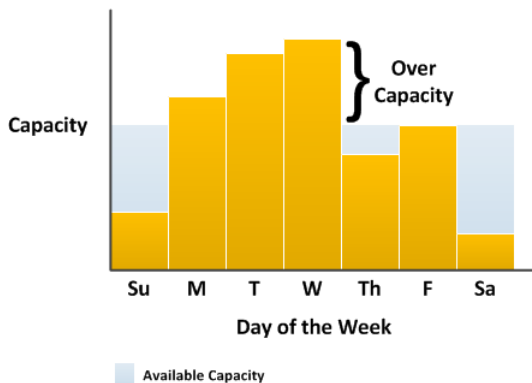
By adding Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. Because Auto Scaling uses EC2 instances, you only have to pay for the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while minimizing expenses.
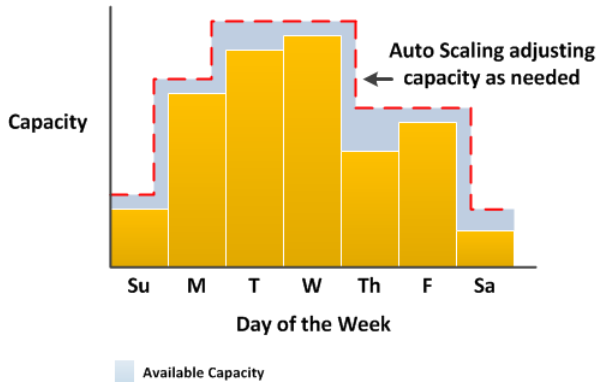
Example: Web App Architecture

In a common web app scenario, you run multiple copies of your app simultaneously to cover the volume of your customer traffic. These multiple copies of your application are hosted on identical EC2 instances (cloud servers), each handling customer requests.
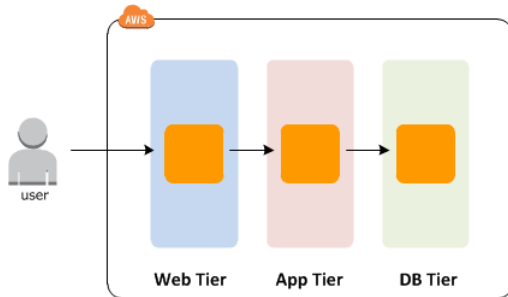
Auto Scaling manages the launch and termination of these EC2 instances on your behalf. You define a set of criteria (such as an Amazon CloudWatch alarm) that determines when the Auto Scaling group launches or terminates EC2 instances. Adding Auto Scaling groups to your network architecture can help you make your application more highly available and fault tolerant.
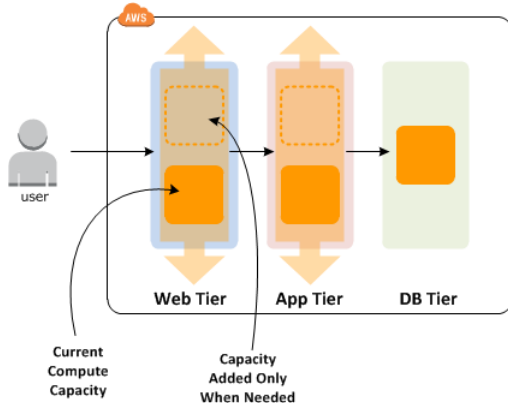
You can create as many Auto Scaling groups as you need. For example, you can create an Auto Scaling group for each tier.

To distribute traffic between the instances in your Auto Scaling groups, you can introduce a load balancer into your architecture. For more information, see Using a Load Balancer With an Auto Scaling Group (p. 46).

Example: Distributing Instances Across Availability Zones

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

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

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

For Auto Scaling groups in a VPC, the EC2 instances are launched in subnets. You can create your VPC with one or more subnets in each Availability Zone. You select the subnets for your EC2 instances when you create or update the Auto Scaling group. For more information, see Launching Auto Scaling Instances in a VPC (p. 26).

Instance Distribution

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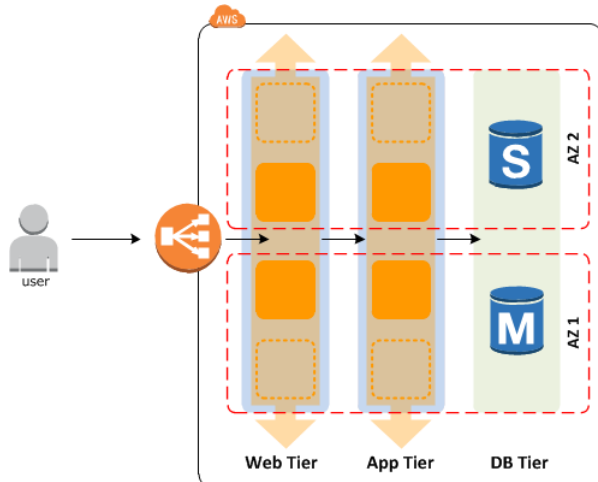
Rebalancing Activities

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

- You issue a request to change the Availability Zones for your group.
- You explicitly call for termination of a specific instance that caused the group to become unbalanced.
- An Availability Zone that previously had insufficient capacity recovers and has additional capacity available.
- An Availability Zone that previously had a Spot market price above your Spot bid price has a decrease that brings its market price below your bid price.

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

Because Auto Scaling attempts to launch new instances before terminating the old ones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities. To avoid this problem, the system can temporarily exceed the specified maximum capacity of a group by a 10 percent margin (or by a 1-instance margin, whichever is greater) during a rebalancing activity. The margin is extended only if the group is at or near maximum capacity and needs rebalancing, either because of user-requested rezoning or to compensate for zone availability issues. The extension lasts only as long as needed to rebalance the group typically a few minutes.

Auto Scaling Lifecycle

The EC2 instances in an Auto Scaling group have a path, or lifecycle, that differs from that of other EC2 instances. The lifecycle starts when the Auto Scaling group launches an instance and puts it into service. The lifecycle ends when you terminate the instance, or the Auto Scaling group takes the instance out of service and terminates it.

Note
You are billed for instances as soon as they are launched, including the time that they are not yet in service.

The following illustration shows the transitions between instance states in the Auto Scaling lifecycle.
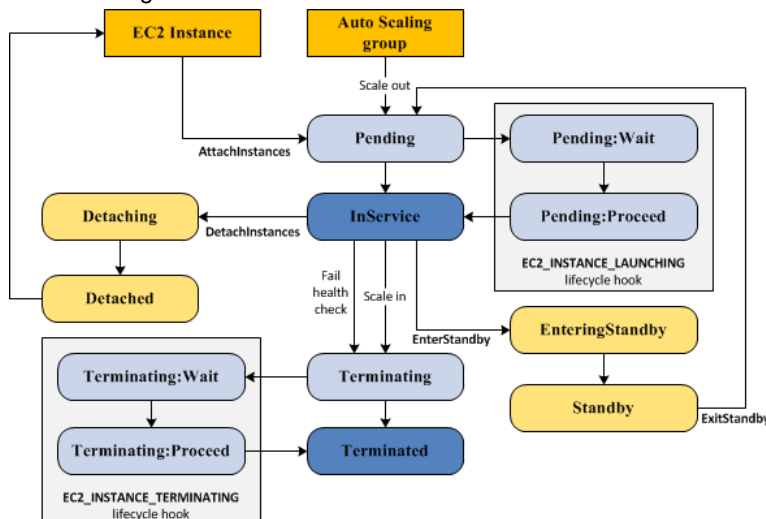
Scale Out

The following scale out events direct the Auto Scaling group to launch EC2 instances and attach them to the group:

When a scale out event occurs, the Auto Scaling group launches the required number of EC2 instances, using its assigned launch configuration. These instances start in the `Pending` state. If you add a lifecycle hook to your Auto Scaling group, you can perform a custom action here. For more information, see Lifecycle Hooks (p. 9).

When each instance is fully configured and passes the Amazon EC2 health checks, it is attached to the Auto Scaling group and it enters the `InService` state. The instance is counted against the desired capacity of the Auto Scaling group.

Instances In Service

Instances remain in the `InService` state until one of the following occurs:

Scale In

It is important that you create a scale in event for each scale out event that you create. This helps ensure that the resources assigned to your application match the demand for those resources as closely as possible.

The following scale in events direct the Auto Scaling group to detach EC2 instances from the group and terminate them:

- You manually decrease the size of the group.
- You create a scaling policy to automatically decrease the size of the group based on a specified decrease in demand.
- You set up scaling by schedule to decrease the size of the group at a specific time.

When a scale in event occurs, the Auto Scaling group detaches one or more instances. The Auto Scaling group uses its termination policy to determine which instances to terminate. Instances that are in the process of detaching from the Auto Scaling group and shutting down enter the `Terminating` state, and can't be put back into service. If you add a lifecycle hook to your Auto Scaling group, you

can perform a custom action here. Finally, the instances are completely terminated and enter the `Terminated` state.

Attach an Instance

You can attach a running EC2 instance that meets certain criteria to your Auto Scaling group. After the instance is attached, it is managed as part of the Auto Scaling group.

For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 61).

Detach an Instance

You can detach an instance from your Auto Scaling group. After the instance is detached, you can manage it separately from the Auto Scaling group or attach it to a different Auto Scaling group.

For more information, see Detach EC2 Instances From Your Auto Scaling Group (p. 65).

Lifecycle Hooks

You can add a lifecycle hook to your Auto Scaling group so that you can perform custom actions when instances launch or terminate.

When Auto Scaling responds to a scale out event, it launches one or more instances. These instances start in the `Pending` state. If you added an `autoscaling:EC2_INSTANCE_LAUNCHING` lifecycle hook to your Auto Scaling group, the instances move from the `Pending` state to the `Pending:Wait` state. After you complete the lifecycle action, the instances enter the `Pending:Proceed` state. When the instances is fully configured, they are attached to the Auto Scaling group and they enter the `InService` state.

When Auto Scaling responds to a scale in event, it terminates one or more instances. These instances are detached from the Auto Scaling group and enter the `Terminating` state. If you added an `autoscaling:EC2_INSTANCE_TERMINATING` lifecycle hook to your Auto Scaling group, the instances move from the `Terminating` state to the `Terminating:Wait` state. After you complete the lifecycle action, the instances enter the `Terminating:Proceed` state. When the instances are fully terminated, they enter the `Terminated` state.

For more information, see Auto Scaling Lifecycle Hooks (p. 89).

Enter and Exit Standby

You can put any instance that is in an `InService` state into a `Standby` state. This enables you to remove the instance from service, troubleshoot or make changes to it, and then put it back into service.

Instances in a `Standby` state continue to be managed by the Auto Scaling group. However, they are not an active part of your application until you put them back into service.

For more information, see Temporarily Removing Instances from Your Auto Scaling Group (p. 95).

Auto Scaling Limits

To view the current limits on your Auto Scaling resources, use the describe-account-limits (AWS CLI) command. To request a limit increase, use the Auto Scaling Limits form.

The following table lists the default limits related to your Auto Scaling resources.

Resource	Default Limit
Launch configurations	100
Auto Scaling groups	20
Scaling policies per Auto Scaling group	50
Scheduled actions per Auto Scaling group	125
Lifecycle hooks per Auto Scaling group	50
SNS topics per Auto Scaling group	10
Classic Load Balancers per Auto Scaling group	50*
Target groups per Auto Scaling group	50*
Step adjustments per scaling policy	20

* Note that you can attach or detach at most 10 at a time.

For information about the limits for other services, see AWS Service Limits in the *Amazon Web Services General Reference*.

Setting Up Auto Scaling

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

Tasks
- Sign Up for AWS (p. 11)
- Prepare to Use Amazon EC2 (p. 11)

Sign Up for AWS

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

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

To sign up for an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

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

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

Prepare to Use Amazon EC2

If you haven't used Amazon EC2 before, complete the tasks described in the Amazon EC2 documentation. For more information, see Setting Up with Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances* or Setting Up with Amazon EC2 in the *Amazon EC2 User Guide for Windows Instances*, depending on which operating system you plan to use for your EC2 instances.

Getting Started with Auto Scaling

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

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

Tasks

Step 1: Create a Launch Configuration

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

To create a launch configuration

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar, select a region. The Auto Scaling resources that you create are tied to the region you specify and are not replicated across regions. For more information, see Example: Distributing Instances Across Availability Zones (p. 6).
3. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
4. On the **Welcome to Auto Scaling** page, choose **Create Auto Scaling group**.
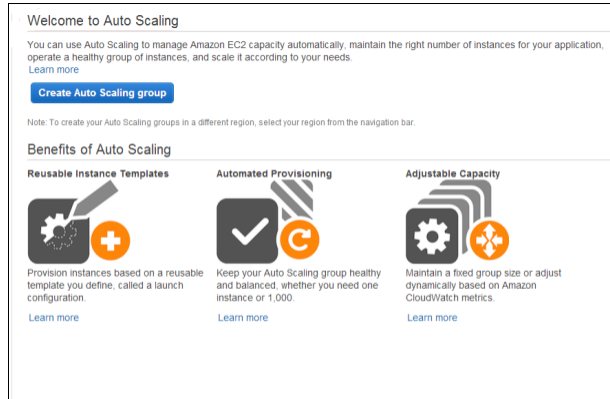
5. On the **Create Auto Scaling Group** page, choose **Create launch configuration**.

6. On the **Choose AMI** page, there is a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instance. Select the 64-bit Amazon Linux AMI.

7. On the **Choose Instance Type** page, select a hardware configuration for your instance. We recommend that you keep the default, a `t2.micro` instance. Choose **Next: Configure details**.

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

8. On the **Configure Details** page, do the following:

   a. For **Name**, type a name for your launch configuration (for example, `my-first-lc`).

   b. For **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to your instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.

   c. Choose **Skip to review**.

9. On the **Review** page, choose **Edit security groups**. Follow the instructions to choose an existing security group, and then choose **Review**.

10. On the **Review** page, choose **Create launch configuration**.

11. On the **Select an existing key pair or create a new key pair** page, select one of the listed options. Note that you won't connect to your instance as part of this tutorial. Therefore, you can select **Proceed without a key pair** unless you intend to connect to your instance.

12. Choose **Create launch configuration**.

Step 2: Create an Auto Scaling Group

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

To create an Auto Scaling group

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

   a. For **Group name**, type a name for your Auto Scaling group (for example, `my-first-asg`.

   b. Keep **Group size** set to the default value of `1` instance for this tutorial.
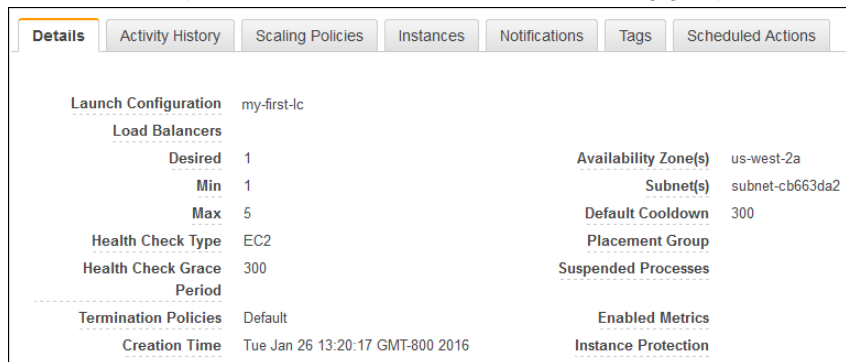
      c.   If you are launching a `t2.micro` instance, you must select a VPC in **Network**. Otherwise, if your account supports EC2-Classic and you are launching a type of instance that doesn't require a VPC, you can select either `Launch into EC2-Classic` or a VPC.

      d.   If you selected a VPC in the previous step, select one or more subnets from **Subnet**. If you selected EC2-Classic in the previous step, select one or more Availability Zones from **Availability Zone(s)**.

      e.   Choose **Next: Configure scaling policies**.

2. On the **Configure scaling policies** page, select **Keep this group at its initial size** and choose **Review**.

3. On the **Review** page, choose **Create Auto Scaling group**.

4. On the **Auto Scaling group creation status** page, choose **Close**.

Step 3: Verify Your Auto Scaling Group

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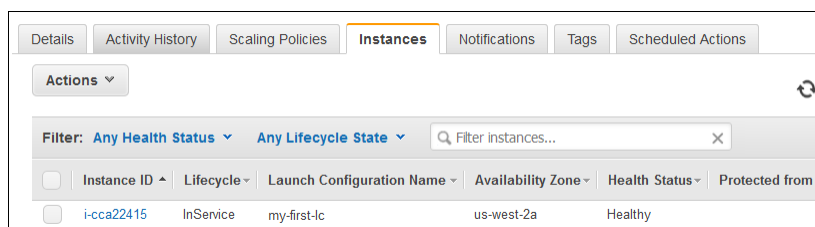
To verify that your Auto Scaling group has launched an EC2 instance

1. On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.

2. The **Details** tab provides information about the Auto Scaling group.

3. On the **Activity History** tab, the **Status** column shows the current status of your instance. While your instance is launching, the status column shows `In progress`. The status changes to `Successful` after the instance is launched. You can also use the refresh button to see the current status of your instance.

4. On the **Instances** tab, the **Lifecycle** column shows the state of your instance. You can see that your Auto Scaling group has launched your EC2 instance, and that it is in the `InService` lifecycle state. The **Health Status** column shows the result of the EC2 instance health check on your instance.

5. (Optional) If you want, you can try the following experiment to learn more about Auto Scaling. The minimum size for your Auto Scaling group is 1 instance. Therefore, if you terminate the running instance, Auto Scaling must launch a new instance to replace it.

a.   On the **Instances** tab, select the ID of the instance. This shows you the instance on the **Instances** page.

b.   Choose **Actions**, **Instance State**, **Terminate**. When prompted for confirmation, choose **Yes, Terminate**.

c.   On the navigation pane, choose **Auto Scaling Groups**, **Activity History**. The default cooldown for the Auto Scaling group is 300 seconds (5 minutes), so it takes about 5 minutes until you see the scaling activity. When the scaling activity starts, you'll see an entry for the termination of the first instance and an entry for the launch of a new instance. The **Instances** tab shows the new instance only.

d.   On the navigation pane, choose **Instances**. This page shows both the terminated instance and the running instance.

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

- Maintaining the Number of Instances in Your Auto Scaling Group (p. 58)
- Manual Scaling (p. 59)
- Dynamic Scaling (p. 70)
- Getting SNS Notifications When Your Auto Scaling Group Scales (p. 116)

Step 4: (Optional) Delete Your Auto Scaling Infrastructure

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

To delete your Auto Scaling group

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2.   On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3.   Select your Auto Scaling group (for example, `my-first-asg`).
4.   Choose **Actions**, **Delete**. When prompted for confirmation, choose **Yes, Delete**.

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

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

To delete your launch configuration

1.   On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
2.   Select your launch configuration (for example, `my-first-lc`).
3.   Choose **Actions**, **Delete launch configuration**. When prompted for confirmation, choose **Yes, Delete**.

Tutorial: Set Up a Scaled and Load-Balanced Application

You can attach a load balancer to your Auto Scaling group. The load balancer automatically distributes incoming traffic across the instances in the group. For more information about the benefits of using Elastic Load Balancing with Auto Scaling, see Using a Load Balancer With an Auto Scaling Group (p. 46).

This tutorial attaches a load balancer to an Auto Scaling group when you create the group. To attach a load balancer to an existing Auto Scaling group, see Attaching a Load Balancer to Your Auto Scaling Group (p. 47).

Contents

Prerequisites

- (Optional) Create an IAM role that grants your application the access to AWS that it needs.
- Launch an instance; be sure to specify the IAM role (if you created one) and specify any configuration scripts that you need as user data. Connect to the instance and customize it. For example, you can install software and applications and copy data. Test your application on your instance to ensure that your instance is configured correctly. Create a custom Amazon Machine Image (AMI) from your instance. You can terminate the instance if you no longer need it.
- Create a load balancer. Elastic Load Balancing supports two types of load balancers: Classic Load Balancers and Application Load Balancers. You can create either type of load balancer to attach to your Auto Scaling group. For more information, see the Elastic Load Balancing User Guide.

  With a Classic Load Balancer, instances are registered with the load balancer, and with an Application Load Balancer, instances are registered as targets with a target group. When you plan to use your load balancer with an Auto Scaling group, you don't need to register your EC2 instances with the load balancer or target group. After you attach a load balancer or target group to your Auto Scaling group, Auto Scaling registers your instances with the load balancer or target group when it launches them.

Configure Scaling and Load Balancing Using the AWS Management Console

Complete the following tasks to set up a scaled and load-balanced application when you create your Auto Scaling group.

Tasks

Create or Select a Launch Configuration

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

To select an existing launch configuration

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar at the top of the screen, select the region that you used when creating your load balancer.
3. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
4. Select a launch configuration.
5. Choose **Create Auto Scaling group**.

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

To create a launch configuration

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar at the top of the screen, select the region that you used when creating your load balancer.
3. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**. If you don't have any Auto Scaling resources, you see a welcome page; choose **Create Auto Scaling group**.
4. Choose **Create launch configuration**.
5. On the **Choose AMI** page, select your custom AMI.
6. On the **Choose Instance Type** page, select a hardware configuration for your instance, and then choose **Next: Configure details**.
7. On the **Configure Details** page, do the following:

    a. For **Name**, type a name for your launch configuration.

    b. (Optional) To securely distribute credentials to your EC2 instance, select your IAM role.

    c. (Optional) If you need to connect to an instance in a nondefault VPC, for **Advanced Details**, **IP Address Type**, choose **Assign a public IP address to every instance**.

    d. (Optional) To specify user data or a configuration script for your instance, for **Advanced Details**, **User data**, paste your configuration script.

    e. Choose **Skip to review**.

8. On the **Review** page, choose **Edit security groups**. Follow the instructions to choose an existing security group, and then choose **Review**.
9. On the **Review** page, choose **Create launch configuration**.

10. On the **Select an existing key pair or create a new key pair** page, select one of the listed options. Select the acknowledgment check box, and then choose **Create launch configuration**.

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

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

Create an Auto Scaling Group

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

To create an Auto Scaling group

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

   a. For **Group name**, type a name for your Auto Scaling group.

   b. For **Group size**, type the initial number of instances for your Auto Scaling group.

   c. If you selected an instance type for your launch configuration that requires a VPC, such as a T2 instance, you must select a VPC for **Network**. Otherwise, if your account supports EC2-Classic and you selected an instance type that doesn't require a VPC, you can select either `Launch into EC2-Classic` or a VPC.

   d. If you selected a VPC in the previous step, select one or more subnets from **Subnet**. If you selected EC2-Classic instead, select one or more Availability Zones from **Availability Zone(s)**.

   e. For **Advanced Details**, select `Receive traffic from Elastic Load Balancer(s)` and then do one of the following:

      - [Classic Load Balancers] Select your load balancer from **Load Balancers**.
      - [Target groups] Select your target group from **Target Groups**.

   f. For **Advanced Details**, select `Receive traffic from Elastic Load Balancer(s)` and then select your load balancer from **Load Balancers**.

   g. (Optional) To use Elastic Load Balancing health checks, choose **ELB** for **Advanced Details**, **Health Check Type**.

   h. Choose **Next: Configure scaling policies**.

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

   If you want to configure scaling policies for your Auto Scaling group, see Scaling Based on Metrics (p. 73).

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

(Optional) Verify that Your Load Balancer is Attached to Your Auto Scaling Group

To verify that your load balancer is attached to your Auto Scaling group

1. Select your Auto Scaling group.

2. On the **Details** tab, **Load Balancers** shows any attached load balancers and **Target Groups** shows any attached target groups.

3. On the **Details** tab, **Load Balancers** shows any attached load balancers.

4. On the **Activity History** tab, the **Status** column shows you the status of your Auto Scaling instances. While an instance is launching, its status is `In progress`. The status changes to `Successful` after the instance is launched.

5. On the **Instances** tab, the **Lifecycle** column shows the state of your Auto Scaling instances. After an instance is ready to receive traffic, its state is `InService`.

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

Configure Scaling and Load Balancing Using the AWS CLI

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

Tasks
- Create a Launch Configuration (p. 19)
- Create an Auto Scaling Group with a Load Balancer (p. 19)

Create a Launch Configuration

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

To create the launch configuration

Use the following create-launch-configuration command:

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

Create an Auto Scaling Group with a Load Balancer

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

To create an Auto Scaling group with an attached Classic Load Balancer

Use the following create-auto-scaling-group command with the --load-balancer-names option to create an Auto Scaling group with an attached Classic Load Balancer:

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

To create an Auto Scaling group with an attached target group

Use the following create-auto-scaling-group command with the --target-group-arns option to create an Auto Scaling group with an attached target group:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-lb-asg
 \
--launch-configuration-name my-lc \
--vpc-zone-identifier "subnet-41767929" \
--vpc-zone-identifier "subnet-b7d581c0" \
--target-group-arns "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/1234567890123456" \
--max-size 5 --min-size 1 --desired-capacity 2
```

Launch Configurations

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

When you create an Auto Scaling group, you must specify a launch configuration. You can specify your launch configuration with multiple Auto Scaling groups. However, you can only specify one launch configuration for an Auto Scaling group at a time, and you can't modify a launch configuration after you've created it. Therefore, if you want to change the launch configuration for your Auto Scaling group, you must create a launch configuration and then update your Auto Scaling group with the new launch configuration. When you change the launch configuration for your Auto Scaling group, any new instances are launched using the new configuration parameters, but existing instances are not affected.

Contents

Creating a Launch Configuration

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

Alternatively, you can create a launch configuration using the attributes from a running EC2 instance. For more information, see Creating a Launch Configuration Using an EC2 Instance (p. 22).

To create a launch configuration using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar at the top of the screen, the current region is displayed. Select a region for your Auto Scaling group that meets your needs.
3. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**. If you are new to Auto Scaling, you see a welcome page; choose **Create Auto Scaling group**.
4. Choose **Create launch configuration**.

5. On the **Choose AMI** page, select an AMI.

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

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

7. On the **Configure Details** page, do the following:

   a. For **Name**, type a name for your launch configuration.

   b. For **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect to your instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.

   c. Choose **Skip to review**.

8. On the **Review** page, choose **Edit security groups**. Follow the instructions to choose an existing security group, and then choose **Review**.

9. On the **Review** page, choose **Create launch configuration**.

10. For **Select an existing key pair or create a new key pair**, select one of the listed options. Select the acknowledgment check box, and then choose **Create launch configuration**.

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

To create a launch configuration using the command line

You can use one of the following commands:

- create-launch-configuration (AWS CLI)
- New-ASLaunchConfiguration (AWS Tools for Windows PowerShell)

Creating a Launch Configuration Using an EC2 Instance

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

> **Tip**
> You can create an Auto Scaling group directly from an EC2 instance (p. 40). When you use this feature, Auto Scaling automatically creates a launch configuration for you as well.

If the specified instance has properties that are not currently supported by Auto Scaling, instances launched by Auto Scaling using the launch configuration created from the identified instance might not be identical to the identified instance.

There are differences between creating a launch configuration from scratch and creating a launch configuration from an existing EC2 instance. When you create a launch configuration from scratch, you specify the image ID, instance type, optional resources (such as storage devices), and optional settings (like monitoring). When you create a launch configuration from a running instance, by default Auto Scaling derives attributes for the launch configuration from the specified instance, plus the block device mapping for the AMI that the instance was launched from (ignoring any additional block devices that were added to the instance after launch).

When you create a launch configuration using a running instance, you can override the following attributes by specifying then as part of the same request: AMI, block devices, key pair, instance profile, instance type, kernel, monitoring, placement tenancy, ramdisk, security groups, Spot price, user data, whether the instance has a public IP address is associated, and whether the instance is EBS-optimized.

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

Examples

Create a Launch Configuration Using an EC2 Instance

To create a launch configuration using the attributes of an existing EC2 instance, specify the ID of the instance.

Important
The AMI used to launch the specified instance must still exist.

Create a Launch Configuration from an EC2 Instance Using the AWS Management Console

You can use the console to create a launch configuration and an Auto Scaling group from a running EC2 instance and add the instance to the new Auto Scaling group. For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 61).

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

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

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance --instance-id i-a8e09d9c
```

You can use the following describe-launch-configurations command to describe the launch configuration and verify that its attributes match those of the instance:

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

The following is an example response:

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
```

Auto Scaling User Guide
Create a Launch Configuration from an
Instance and Override the Block Devices

```
        "LaunchConfigurationARN": "arn",
        "InstanceMonitoring": {
            "Enabled": false
        },
        "ImageId": "ami-05355a6c",
        "CreatedTime": "2014-12-29T16:14:50.382Z",
        "BlockDeviceMappings": [],
        "KeyName": "my-key-pair",
        "SecurityGroups": [
            "sg-8422d1eb"
        ],
        "LaunchConfigurationName": "my-lc-from-instance",
        "KernelId": "null",
        "RamdiskId": null,
        "InstanceType": "t1.micro",
        "AssociatePublicIpAddress": true
    }
  ]
}
```

Create a Launch Configuration from an Instance and Override the Block Devices

By default, Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration, except that the block devices come from the AMI used to launch the instance, not the instance. To add block devices to the launch configuration, override the block device mapping for the launch configuration.

Important
The AMI used to launch the specified instance must still exist.

Create a Launch Configuration and Override the Block Devices Using the AWS CLI

Use the following create-launch-configuration command to create a launch configuration using an EC2 instance but with a custom block device mapping:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-
lc-from-instance-bdm --instance-id i-a8e09d9c
--block-device-mappings "[{\"DeviceName\":\"/dev/sda1\",\"Ebs\":{\"SnapshotId
\":\"snap-3decf207\"}},{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":
\"snap-eed6ac86\"}}]"
```

Use the following describe-launch-configurations command to describe the launch configuration and verify that it uses your custom block device mapping:

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

The following example response describes the launch configuration:

```
{
    "LaunchConfigurations": [
        {
```

```
        "UserData": null,
        "EbsOptimized": false,
        "LaunchConfigurationARN": "arn",
        "InstanceMonitoring": {
            "Enabled": false
        },
        "ImageId": "ami-c49c0dac",
        "CreatedTime": "2015-01-07T14:51:26.065Z",
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/sda1",
                "Ebs": {
                    "SnapshotId": "snap-3decf207"
                }
            },
            {
                "DeviceName": "/dev/sdf",
                "Ebs": {
                    "SnapshotId": "snap-eed6ac86"
                }
            }
        ],
        "KeyName": "my-key-pair",
        "SecurityGroups": [
            "sg-8637d3e3"
        ],
        "LaunchConfigurationName": "my-lc-from-instance-bdm",
        "KernelId": null,
        "RamdiskId": null,
        "InstanceType": "t1.micro",
        "AssociatePublicIpAddress": true
        }
    ]
}
```

Create a Launch Configuration and Override the Instance Type

By default, Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration. Depending on your requirements, you might want to change some of these attributes. Auto Scaling provides you with options to override attributes from the instance and use the values that you need. For example, you can override the instance type.

Important
The AMI used to launch the specified instance must still exist.

Create a Launch Configuration and Override the Instance Type Using the AWS CLI

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

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

Use the following describe-launch-configurations command to describe the launch configuration and verify that the instance type was overridden:

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

The following example response describes the launch configuration:

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-changetype",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "m1.small",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

Launching Auto Scaling Instances in a VPC

Amazon Virtual Private Cloud (Amazon VPC) enables you to define a virtual networking environment in a private, isolated section of the AWS cloud. You have complete control over your virtual networking environment. For more information, see the *Amazon VPC User Guide*.

Within a virtual private cloud (VPC), you can launch AWS resources such as an Auto Scaling group. An Auto Scaling group in a VPC works essentially the same way as it does on Amazon EC2 and supports the same set of features.

A subnet in Amazon VPC is a subdivision within an Availability Zone defined by a segment of the IP address range of the VPC. Using subnets, you can group your instances based on your security and operational needs. A subnet resides entirely within the Availability Zone it was created in. You launch Auto Scaling instances within the subnets.

To enable communication between the Internet and the instances in your subnets, you must create an Internet gateway and attach it to your VPC. An Internet gateway enables your resources within the subnets to connect to the Internet through the Amazon EC2 network edge. If a subnet's traffic is routed to an Internet gateway, the subnet is known as a *public* subnet. If a subnet's traffic is not routed to an Internet gateway, the subnet is known as a *private* subnet. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that need not be connected to the Internet.

Prerequisites

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

Contents

Default VPC

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

For information about default VPCs and checking whether your account comes with a default VPC, see Your Default VPC and Subnets in the *Amazon VPC Developer Guide*.

IP Addressing in a VPC

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

Instance Placement Tenancy

Dedicated Instances are physically isolated at the host hardware level from instances that aren't dedicated and from instances that belong to other AWS accounts. When you create a VPC, by default its tenancy attribute is set to `default`. In such a VPC, you can launch instances with a tenancy value of `dedicated` so that they run as single-tenancy instances. Otherwise, they run as shared-tenancy instances by default. If you set the tenancy attribute of a VPC to `dedicated`, all instances launched in the VPC run as single-tenancy instances. For more information, see Dedicated Instances in the *Amazon VPC User Guide*. For pricing information, see the Amazon EC2 Dedicated Instances product page.

When you create a launch configuration, the default value for the instance placement tenancy is `null` and the instance tenancy is controlled by the tenancy attribute of the VPC. The following table summarizes the instance placement tenancy of the Auto Scaling instances launched in a VPC.

Launch Configuration Tenancy	VPC Tenancy = default	VPC Tenancy = dedicated
not specified	shared-tenancy instance	Dedicated Instance
default	shared-tenancy instance	Dedicated Instance

Launch Configuration Tenancy	VPC Tenancy = default	VPC Tenancy = dedicated
dedicated	Dedicated Instance	Dedicated Instance

You can specify the instance placement tenancy for your launch configuration as `default` or `dedicated` using the create-launch-configuration command with the `--placement-tenancy` option. For example, the following command sets the launch configuration tenancy to `dedicated`:

```
aws autoscaling create-launch-configuration --launch-configuration-name my-launch-config --placement-tenancy dedicated --image-id ...
```

You can use the following describe-launch-configurations command to verify the instance placement tenancy of the launch configuration:

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-launch-config
```

The following is example output for a launch configuration that creates Dedicated Instances. Note that `PlacementTenancy` is not part of the output for this command unless you have explicitly set the instance placement tenancy.

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

Linking EC2-Classic Instances to a VPC

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

If you have running EC2-Classic instances in your Auto Scaling group, you can link them to a VPC with ClassicLink enabled. For more information, see Linking an Instance to a VPC in the *Amazon EC2*

User Guide for Linux Instances. Alternatively, you can update the Auto Scaling group to use a launch configuration that automatically links the EC2-Classic instances to a VPC at launch, then terminate the running instances and let Auto Scaling launch new instances that are linked to the VPC.

Link to a VPC Using the AWS Management Console

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

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

1. Verify that the VPC has ClassicLink enabled. For more information, see Viewing Your ClassicLink-Enabled VPCs in the *Amazon EC2 User Guide for Linux Instances*.
2. Create a security group for the VPC that you are going to link EC2-Classic instances to, with rules to control communication between the linked EC2-Classic instances and instances in the VPC.
3. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
4. On the navigation pane, choose **Launch Configurations**. If you are new to Auto Scaling, you see a welcome page. Choose **Create Auto Scaling group**.
5. Choose **Create launch configuration**.
6. On the **Choose AMI** page, select an AMI.
7. On the **Choose an Instance Type** page, select an instance type, and then choose **Next: Configure details**.
8. On the **Configure details** page, do the following:

   a. Type a name for your launch configuration.
   b. Expand **Advanced Details**, select the **IP Address Type** that you need, and then select **Link to VPC**.
   c. For **VPC**, select the VPC with ClassicLink enabled from step 1.
   d. For **Security Groups**, select the security group from step 2.
   e. Choose **Skip to review**.

9. On the **Review** page, make any changes that you need, and then choose **Create launch configuration**. For **Select an existing key pair or create a new key pair**, select an option, select the acknowledgment check box (if present), and then choose **Create launch configuration**.
10. When prompted, follow the directions to create an Auto Scaling group that uses the new launch configuration. Be sure to select **Launch into EC2-Classic** for **Network**. Otherwise, choose **Cancel** and then add your launch configuration to an existing Auto Scaling group as follows:

    a. On the navigation pane, choose **Auto Scaling Groups**.
    b. Select your Auto Scaling group, choose **Actions**, **Edit**.
    c. For **Launch Configuration**, select your new launch configuration and then choose **Save**.

Link to a VPC Using the AWS CLI

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

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

1. Verify that the VPC has ClassicLink enabled. For more information, see Viewing Your ClassicLink-Enabled VPCs in the *Amazon EC2 User Guide for Linux Instances*.
2. Create a security group for the VPC that you are going to link EC2-Classic instances to, with rules to control communication between the linked EC2-Classic instances and instances in the VPC.

3. Create a launch configuration using the create-launch-configuration command as follows, where *vpd_id* is the ID of the VPC with ClassicLink enabled from step 1 and *group_id* is the security group from step 2:

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

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

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

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

Examples

For examples, see the following tutorials:

- Getting Started with Auto Scaling (p. 12)
- Hosting a Web App on Amazon Web Services
- Hosting a .NET Web App on Amazon Web Services

Launching Spot Instances in Your Auto Scaling Group

Spot instances are a cost-effective choice compared to On-Demand instances, if you can be flexible about when your applications run and if your applications can be interrupted. You can set up Auto Scaling to launch Spot instances instead of On-Demand instances.

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

Here's how Spot instances work with Auto Scaling:

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

- **Maintaining your Spot instances.** When your Spot instance is terminated, Auto Scaling attempts to launch a replacement instance to maintain the desired capacity for the group. If the bid price is higher than the market price, then it launches a Spot instance. Otherwise, Auto Scaling keeps trying.
- **Balancing across Availability Zones.** If you specify multiple Availability Zones, Auto Scaling distributes the bids across these Availability Zones. If your Spot bid price is too low in one Availability Zone for any bids to be fulfilled, Auto Scaling checks whether bids were fulfilled in the other Availability Zones. If so, Auto Scaling cancels the bids that failed and redistributes them across the Availability Zones that have bids fulfilled. If the price in an Availability Zone with no fulfilled bids drops enough that future bids succeed, Auto Scaling rebalances across all the Availability Zones. For more information, see Rebalancing Activities (p. 7).
- **Auto Scaling and Spot instance termination.** Auto Scaling can terminate or replace Spot instances just as it can terminate or replace On-Demand instances. For more information, see Controlling Which Instances Auto Scaling Terminates During Scale In (p. 84).

Contents

Launching Spot Instances Using the AWS Management Console

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

Tasks

Create a Launch Configuration

To create a launch configuration

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

    a. For **Name**, type a name for your launch configuration. Consider including "Spot" and the bid price in this name.

    b. Select **Request Spot Instances**. When you select this option, you'll see the current prices for the Availability Zones in the region. For **Maximum price**, type your bid price.

    c. For **Advanced Details**, select an IP address type. If you want to connect to an instance in a VPC, you must select an option that assigns a public IP address. If you want to connect

to your instance but aren't sure whether you have a default VPC, select **Assign a public IP address to every instance**.

d.   Choose **Skip to review**.

7.   On the **Review** page, choose **Edit security groups**. Follow the instructions to choose an existing security group, and then choose **Review**.

8.   On the **Review** page, notice that the launch configuration details include your bid price. Choose **Create launch configuration**.

9.   On the **Select an existing key pair or create a new key pair** page, select one of the listed options. Select the acknowledgment check box, and then choose **Create launch configuration**.

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

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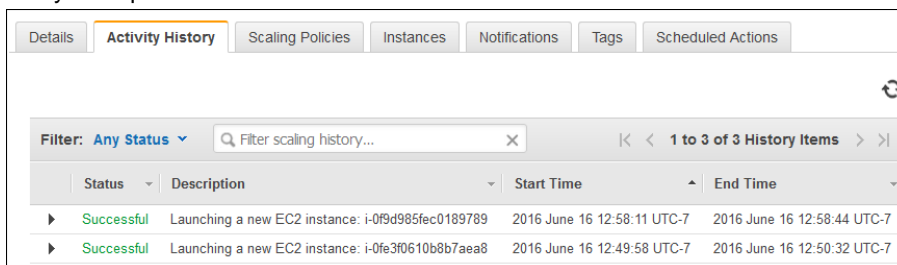
When you create your Auto Scaling group, specify the launch configuration that you just created. Remember that the launch configuration is a template for your instances, and it includes the bid price for your Spot instances.

For more information and directions for creating your Auto Scaling group using the AWS Management Console, see Creating an Auto Scaling Group (p. 39).

Verify and Check Your Instances

To confirm that Auto Scaling is launching your Spot instances

1.   Select your new Auto Scaling group.

2.   On the **Activity History** tab, it shows that Auto Scaling successfully launched the Spot instances that you requested.
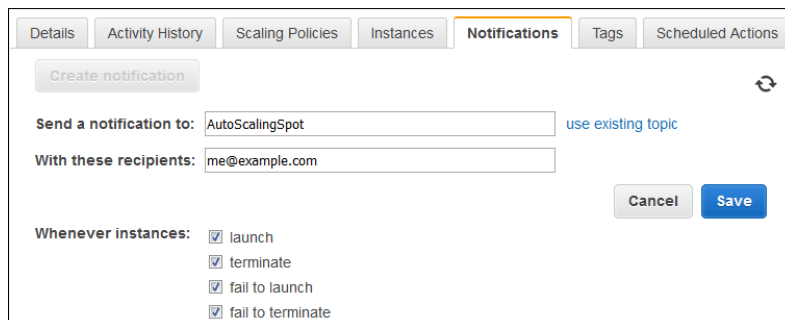
3.   On the **Instances** tab, it shows details about your Spot instances. You'll see that Auto Scaling is launching the instances you requested in the Availability Zones that you specified.

(Optional) Get Notifications When the Auto Scaling Group Changes

To set up notifications

1.   Select the Auto Scaling group.

2.   On the **Notifications** tab, choose **Create notification**.

3.   Choose **create topic**, specify the following, and then choose **Save**:

- **Send a notification to** - AutoScalingSpot
- **With these recipients** - *your email account*

- **Whenever instances** - One or more of launch, terminate, fail to launch, and fail to terminate

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

(Optional) Update the Bid Price

To update the bid price for the Spot instances

1. Create a launch configuration with the same specifications as in Create a Launch Configuration (p. 31), but with a different name and maximum price.
2. Select your Auto Scaling group.
3. On the **Details** tab, choose **Edit**.
4. Select the launch configuration that you just created, and then choose **Save**.

Clean Up

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

To clean up Auto Scaling group and instances

1. Select your Auto Scaling group.
2. Choose **Actions**, **Delete**.
3. When prompted for confirmation, choose **Yes, Delete**.

Launching Spot Instances Using the AWS CLI

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

Tasks

Create a Launch Configuration

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

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

Create an Auto Scaling Group

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

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

Verify and Check Your Instances

Use the describe-scaling-activities command as follows to list the activities that Auto Scaling performed for your Auto Scaling group:

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

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

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

```
            "Cause": "At 2015-03-01T16:21:40Z a difference between desired
 and actual capacity changing the desired capacity, increasing the capacity
 from 0 to 2.",
            "StatusCode": "Successful"
        }
    ]
}
```

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

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

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

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

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

```
            "CreatedTime": "2015-03-01T16:12:35.608Z",
            "AvailabilityZones": [
                "us-west-2b",
                "us-west-2a"
            ],
            "HealthCheckType": "EC2"
        }
    ]
}
```

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

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

The following is example output:

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

(Optional) Get Notifications When the Auto Scaling Group Changes

For information about setting up email notifications in Auto Scaling, see Getting SNS Notifications When Your Auto Scaling Group Scales (p. 116).

(Optional) Update the Bid Price for the Spot Instances

To update the bid price for Spot instances

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

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

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

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

3. View your changes using the describe-scaling-activities command as follows:

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

Clean Up

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

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

Auto Scaling Groups

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

An Auto Scaling group starts by launching enough EC2 instances to meet its desired capacity. The Auto Scaling group maintains this number of instances by performing periodic health checks on the instances in the group. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it. For more information about health check replacements, see Maintaining the Number of Instances in Your Auto Scaling Group (p. 58).

You can use scaling policies to increase or decrease the number of running EC2 instances in your group automatically to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group and launches or terminates the instances as needed. If you manually scale or scale on a schedule, you must adjust the desired capacity of the group in order for the changes to take effect. For more information, see Scaling the Size of Your Auto Scaling Group (p. 56).

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

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

The better you understand your application, the more effective you can make your Auto Scaling architecture.

Contents

Creating an Auto Scaling Group

When you create an Auto Scaling group, you must specify the launch configuration to use for launching the instances, and the minimum number of instances your group must maintain at all times. To get the most out of your Auto Scaling group, you should also specify the following:

- Desired capacity. This parameter specifies the number of instances that you'd like to have in the Auto Scaling group. If you don't specify a desired capacity, the default desired capacity is the minimum number of instances that you specified.
- Availability Zones or subnets. It is often a good idea to build or modify your applications in AWS to use more than one Availability Zone. If your Auto Scaling group operates within a VPC, you can alternatively specify which subnets you want Auto Scaling to use.
- Metrics and health checks. An effective Auto Scaling group uses metrics to determine when it should launch or terminate instances. In addition, it's helpful to define health checks which Auto Scaling uses to determine if an instance is healthy or, if not, if Auto Scaling should terminate the instance and replace it.

Alternatively, you can create an Auto Scaling group using an EC2 instance instead of a launch configuration. For more information, see Creating an Auto Scaling Group Using an EC2 Instance (p. 40).

Prerequisites

Create a launch configuration. For more information, see Creating a Launch Configuration (p. 21).

To create an Auto Scaling group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar at the top of the screen, select the same region that you used when you created the launch configuration.
3. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
4. Choose **Create Auto Scaling group**.
5. On the **Create Auto Scaling Group** page, select **Create an Auto Scaling group from an existing launch configuration**, select a launch configuration, and then choose **Next Step**.

    Note
    If you do not have any launch configurations, you're first prompted to create one before you can continue with the steps to create an Auto Scaling group.

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

    a. For **Group name**, type a name for your Auto Scaling group.

    b. For **Group size**, type the initial number of instances for your Auto Scaling group.

    c. If you selected an instance type for your launch configuration that requires a VPC, such as a T2 instance, you must select a VPC for **Network**. Otherwise, if your account supports EC2-

Classic and you selected an instance type that doesn't require a VPC, you can choose either `Launch into EC2-Classic` or a VPC.

   d.   If you selected a VPC in the previous step, select one or more subnets from **Subnet**. If you selected EC2-Classic instead, select one or more Availability Zones from **Availability Zone(s)**.

   e.   Choose **Next: Configure scaling policies**.

7.   On the **Configure scaling policies** page, select one of the following options, and then choose **Review**:

   • To manually adjust the size of the Auto Scaling group as needed, select **Keep this group at its initial size**. For more information, see Manual Scaling (p. 59).

   • To automatically adjust the size of the Auto Scaling group based on criteria that you specify, select **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 74).

8.   (Optional) To add tags now, choose **Edit tags** and complete the following steps. Alternatively, you can add tags later on. For more information, see Tagging Auto Scaling Groups and Instances (p. 43).

   a.   For **Key** and **Value**, type the key and the value for your first tag.

   b.   Keep **Tag New Instances** selected if you want Auto Scaling to propagate the tag to the instances launched by your Auto Scaling group.

   c.   Choose **Add tag** to add additional tags, and then type keys and values for the tags.

   d.   Choose **Review**.

9.   On the **Review** page, choose **Create Auto Scaling group**.

10.   On the **Auto Scaling group creation status** page, choose **Close**.

To create an Auto Scaling group using the command line

You can use one of the following commands:

• create-auto-scaling-group (AWS CLI)
• New-ASAutoScalingGroup (AWS Tools for Windows PowerShell)

Creating an Auto Scaling Group Using an EC2 Instance

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

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

Limitations

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

• If the identified instance has tags, the tags are not copied to the `Tags` attribute of the new Auto Scaling group.
• The Auto Scaling group includes the block device mapping from the AMI used to launch the instance; it does not include any block devices attached after instance launch.

Auto Scaling User Guide
Create an Auto Scaling Group from
an EC2 Instance Using the Console

- If the identified instance is registered with one or more load balancers, the load balancer names are not copied to the `LoadBalancerNames` attribute of the new Auto Scaling group.

Prerequisites

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

The EC2 instance must meet the following criteria:

- The instance is in the Availability Zone in which you want to create the Auto Scaling group.
- The instance is not a member of another Auto Scaling group.
- The instance is in `running` state.
- The AMI used to launch the instance must still exist.

Contents

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

You can use the console to create an Auto Scaling group from a running EC2 instance and add the instance to the new Auto Scaling group. For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 61).

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

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

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

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

The following example response shows that the desired capacity of the group is 2, the group has 2 running instances, and the launch configuration is also named *my-asg-from-instance*:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "HealthCheckGracePeriod": 0,
            "SuspendedProcesses": [],
            "DesiredCapacity": 2,
            "Tags": [],
```

```
        "EnabledMetrics": [],
        "LoadBalancerNames": [],
        "AutoScalingGroupName": "my-asg-from-instance",
        "DefaultCooldown": 300,
        "MinSize": 1,
        "Instances": [
            {
                "InstanceId": "i-6bd79d87",
                "AvailabilityZone": "us-west-2a",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-asg-from-instance"
            },
            {
                "InstanceId": "i-6cd79d80",
                "AvailabilityZone": "us-west-2a",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-asg-from-instance"
            }
        ],
        "MaxSize": 2,
        "VPCZoneIdentifier": "subnet-6bea5f06",
        "TerminationPolicies": [
            "Default"
        ],
        "LaunchConfigurationName": "my-asg-from-instance",
        "CreatedTime": "2014-12-29T16:14:50.397Z",
        "AvailabilityZones": [
            "us-west-2a"
        ],
        "HealthCheckType": "EC2"
    }
  ]
}
```

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

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

Creating an Auto Scaling Group Using the Amazon EC2 Launch Wizard

You can create a launch configuration and an Auto Scaling group in a single procedure by using the Amazon EC2 launch wizard. This is useful if you're launching more than one instance, and want to create a new launch configuration and Auto Scaling group from settings you've already selected in the Amazon EC2 launch wizard. You cannot use this option to create an Auto Scaling group using an existing launch configuration.

To create a launch configuration and Auto Scaling group from the launch wizard

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. From the dashboard, choose **Launch Instance**.

3. Choose an AMI, then choose an instance type on the next page, and then choose **Next: Configure Instance Details**.

4. In **Number of instances**, enter the number of instances that you want to launch, and then choose **Launch into Auto Scaling Group**. You do not need to enter any other configuration details on the page.

5. On the confirmation page, choose **Create Launch Configuration**.

6. You are switched to step 3 of the launch configuration wizard. The AMI and instance type are already selected based on the selection you made in the Amazon EC2 launch wizard. Enter a name for the launch configuration, configure any other settings as required, and then choose **Next: Add Storage**.

7. Configure any additional volumes, and then choose **Next: Configure Security Group**.

8. Create a new security group, or choose an existing group, and then choose **Review**.

9. Review the details of the launch configuration, and then choose **Create launch configuration** to choose a key pair and create the launch configuration.

10. On the **Configure Auto Scaling group details** page, the launch configuration you created is already selected for you, and the number of instances you specified in the Amazon EC2 launch wizard is populated for **Group size**. Enter a name for the group, specify a VPC and subnet (if required), and then choose **Next: Configure scaling policies**.

11. On the **Configure scaling policies** page, choose one of the following options, and then choose **Review**:

   - To manually adjust the size of the Auto Scaling group as needed, select **Keep this group at its initial size**. For more information, see Manual Scaling (p. 59).

   - To automatically adjust the size of the Auto Scaling group based on criteria that you specify, select **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 74).

12. On the **Review** page, you can optionally add tags or notifications, and edit other configuration details. When you have finished, choose **Create Auto Scaling group**.

Tagging Auto Scaling Groups and Instances

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

You can specify that Auto Scaling also adds the tags for your Auto Scaling groups to the EC2 instances that it launches. Auto Scaling applies the tags while the instances are in the `Pending` state. Note that if you have a lifecycle hook, the tags are available when the instance enters the `Pending:Wait` state.

Tagging your EC2 instances enables you to see instance cost allocation by tag in your AWS bill. For more information, see Using Cost Allocation Tags in the *AWS Billing and Cost Management User Guide*.

Contents

- Delete Tags (p. 46)

Tag Restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 10.
- The maximum number of tags that you can add or remove using a single call is 25.
- The maximum key length is 127 Unicode characters.
- The maximum value length is 255 Unicode characters.
- Tag keys and values are case sensitive.
- Do not use the `aws:` prefix in your tag names or values, because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix, and they do not count against toward your limit of tags per Auto Scaling group.

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

You can create and assign tags to your Auto Scaling group when you either create or update your Auto Scaling group. You can remove Auto Scaling group tags at any time. For information about assigning tags when you create your Auto Scaling group, see Step 2: Create an Auto Scaling Group (p. 13).

Add or Modify Tags for Your Auto Scaling Group

When you add a tag to your Auto Scaling group, you can specify whether it should be added to instances launched in your Auto Scaling group. If you modify a tag, the updated version of the tag is added to instances launched in the Auto Scaling group after the change. If you create or modify a tag for an Auto Scaling group, these changes are not made to instances that are already running in the Auto Scaling group.

Contents
- Add or Modify Tags Using the AWS Management Console (p. 44)
- Add or Modify Tags Using the AWS CLI (p. 45)

Add or Modify Tags Using the AWS Management Console

Use the Amazon EC2 console to add or modify tags.

To add or modify tags

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Tags** tab, choose **Add/Edit tags**. The **Add/Edit Auto Scaling Group Tags** page lists any existing tags for the Auto Scaling group.
5. To modify existing tags, edit **Key** and **Value**.
6. To add a new tag, choose **Add tag** and edit **Key** and **Value**. You can keep **Tag New Instances** selected to add the tag to the instances launched in the Auto Scaling group automatically, and deselect it otherwise.
7. When you have finished adding tags, choose **Save**.

Add or Modify Tags Using the AWS CLI

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

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

The following is an example response:

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

Use the following describe-tags command to list the tags for the specified Auto Scaling group.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

The following is an example response:

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "test",
            "Key": "environment"
        }
    ]
}
```

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

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

The following is an example response:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "HealthCheckGracePeriod": 0,
            "SuspendedProcesses": [],
            "DesiredCapacity": 1,
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "test",
```

```
                    "Key": "environment"
                }
            ],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            ...
        }
    ]
}
```

Delete Tags

You can delete a tag associated with your Auto Scaling group at any time.

Contents

Delete Tags Using the AWS Management Console

To delete a tag using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Tags** tab, choose **Add/Edit tags**. The **Add/Edit Auto Scaling Group Tags** page lists any existing tags for the Auto Scaling group.
5. Choose the delete icon next to the tag.
6. Choose **Save**.

Delete Tags Using the AWS CLI

Use the delete-tags command to delete a tag. For example, the following command deletes a tag with a key of "environment".

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-
scaling-group,Key=environment"
```

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

Using a Load Balancer With an Auto Scaling Group

When you use Auto Scaling, you can automatically increase the size of your Auto Scaling group when demand goes up and decrease it when demand goes down. As Auto Scaling adds and removes EC2 instances, you must ensure that the traffic for your application is distributed across all of your EC2 instances. The Elastic Load Balancing service automatically routes incoming web traffic across such a

dynamically changing number of EC2 instances. Your load balancer acts as a single point of contact for all incoming traffic to the instances in your Auto Scaling group. For more information, see the Elastic Load Balancing User Guide.

To use a load balancer with your Auto Scaling group, create the load balancer and then attach it to the group.

Contents

Attaching a Load Balancer to Your Auto Scaling Group

Auto Scaling integrates with Elastic Load Balancing to enable you to attach one or more load balancers to an existing Auto Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances. To use an Elastic Load Balancing health check with your instances to ensure that traffic is routed only to the healthy instances, see Adding Health Checks to Your Auto Scaling Group (p. 49).

When you attach a load balancer, it enters the `Adding` state while registering the instances in the group. After all instances in the group are registered with the load balancer, it enters the `Added` state. After at least one registered instance passes the health checks, it enters the `InService` state. After the load balancer enters the `InService` state, Auto Scaling can terminate and replace any instances that are reported as unhealthy. Note that if no registered instances pass the health checks (for example, due to a misconfigured health check), the load balancer doesn't enter the `InService` state, so Auto Scaling wouldn't terminate and replace the instances.

When you detach a load balancer, it enters the `Removing` state while deregistering the instances in the group. Note that the instances remain running after they are deregistered. If connection draining is enabled, Elastic Load Balancing waits for in-flight requests to complete or for the maximum timeout to expire (whichever comes first) before deregistering the instances. Note that connection draining is always enabled for Application Load Balancers but must be enabled for Classic Load Balancers. For more information, see Connection Draining in the *Classic Load Balancer Guide*.

Elastic Load Balancing sends data about your load balancers and EC2 instances to Amazon CloudWatch. CloudWatch collects performance data for your resources and presents it as metrics. For more information, see Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch (p. 105). After you attach a load balancer to your Auto Scaling group, you can create scaling policies that use Elastic Load Balancing metrics to scale your application automatically. For more information, see Scaling Based on Metrics (p. 73).

Contents

Prerequisites

Before you begin, create a load balancer in the same region as the Auto Scaling group. Elastic Load Balancing supports two types of load balancers: Classic Load Balancers and Application Load Balancers. You can create either type of load balancer to attach to your Auto Scaling group. For more information, see the Elastic Load Balancing User Guide.

With a Classic Load Balancer, instances are registered with the load balancer, and with an Application Load Balancer, instances are registered as targets with a target group. When you plan to use your load balancer with an Auto Scaling group, you don't need to register your EC2 instances with the load balancer or target group. After you attach a load balancer or target group to your Auto Scaling group, Auto Scaling registers your instances with the load balancer or target group when it launches them.

Add a Load Balancer Using the Console

Use the following procedure to attach a load balancer to an existing Auto Scaling group. To attach your load balancer to your Auto Scaling group when you create the Auto Scaling group, see Tutorial: Set Up a Scaled and Load-Balanced Application (p. 16).

To attach a load balancer to a group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your group.
4. On the **Details** tab, choose **Edit**.
5. Do one of the following:

    a. [Classic Load Balancer] For **Load Balancers**, select your load balancer.
    b. [Application Load Balancer] For **Target Groups**, select your target group.
6. Choose **Save**.

When you no longer need the load balancer, use the following procedure to detach it from your Auto Scaling group.

To detach a load balancer from a group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your group.
4. On the **Details** tab, choose **Edit**.
5. Do one of the following:

    a. [Classic Load Balancer] For **Load Balancers**, remove the load balancer.
    b. [Application Load Balancer] For **Target Groups**, remove the target group.
6. Choose **Save**.

Add a Load Balancer Using the AWS CLI

To attach a Classic Load Balancer

Use the following attach-load-balancers command to attach the specified load balancer to your Auto Scaling group:

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg --
load-balancer-names my-lb
```

To attach a target group

Use the following attach-load-balancer-target-groups command to attach the specified target group to your Auto Scaling group:

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-
name my-asg --target-group-arns my-targetgroup-arn
```

To detach a Classic Load Balancer

Use the following detach-load-balancers command to detach a load balancer from your Auto Scaling group if you no longer need it:

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg --
load-balancer-names my-lb
```

To detach a target group

Use the following detach-load-balancer-target-groups command to detach a target group from your Auto Scaling group if you no longer need it:

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-
name my-asg --target-group-arns my-targetgroup-arn
```

Adding Health Checks to Your Auto Scaling Group

By default, an Auto Scaling group determines the health state of each instance by periodically checking the results of the EC2 instance status checks. If an instance fails the EC2 instance status checks, Auto Scaling considers the instance unhealthy and replaces it. However, if you have attached one or more load balancers to your Auto Scaling group and an instance fails the load balancer health checks, Auto Scaling does not replace the instance by default.

You can configure your Auto Scaling group to use both EC2 instance status checks and load balancer health checks to determine the health status of your instances. If you enable load balancer health checks and an instance fails the health checks, Auto Scaling considers the instance unhealthy and replaces it. If you attach multiple load balancers to an Auto Scaling group, all the load balancers must report that the instance passed the health checks in order for Auto Scaling to consider the instance healthy. If one load balancer reports an instance as unhealthy, Auto Scaling replaces the instance, even if the other load balancers report it as healthy. For more information, see Health Checks for Auto Scaling Instances (p. 104).

If connection draining is enabled for your load balancer, Auto Scaling waits for the in-flight requests to complete or for the maximum timeout to expire, whichever comes first, before terminating instances due to a scaling event or health check replacement. For more information, see Connection Draining in the *Classic Load Balancer Guide*.

Contents

Adding Health Checks Using the Console

Use the following procedure to add an ELB health check with a grace period of 300 seconds to an Auto Scaling group with an attached load balancer.

To add health checks using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3. Select your group.
4. On the **Details** tab, choose **Edit**.
5. For **Health Check Type**, select `ELB`.
6. For **Health Check Grace Period**, enter `300`.
7. Choose **Save**.
8. On the **Instances** tab, the **Health Status** column displays the results of the newly added health checks.

Adding Health Checks Using the AWS CLI

Use the following update-auto-scaling-group command to create a health check with a grace period of 300 seconds:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-lb-asg
 --health-check-type ELB --health-check-grace-period 300
```

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

You can take advantage of the safety and reliability of geographic redundancy by spanning your Auto Scaling group across multiple Availability Zones within a region and then attaching a load balancer to distribute incoming traffic across those Availability Zones. Incoming traffic is distributed equally across all Availability Zones enabled for your load balancer.

> **Note**
> An Auto Scaling group can contain EC2 instances from multiple Availability Zones within the same region. However, an Auto Scaling group can't contain EC2 instances from multiple regions.

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

You can expand the availability of your scaled and load-balanced application by adding an 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.

Contents

Add an Availability Zone Using the Console

Use the following procedure to expand your Auto Scaling group to an additional subnet (EC2-VPC) or Availability Zone (EC2-Classic).

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3. Select your group.

4. On the **Details** tab, choose **Edit**.

5. Do one of the following:

    • [EC2-VPC] In **Subnet(s)**, select the subnet corresponding to the Availability Zone.

    • [EC2-Classic] In **Availability Zones(s)**, select the Availability Zone.

6. Choose **Save**.

7. On the navigation pane, under **NETWORK & SECURITY**, choose **Load Balancers**.

8. Select your load balancer.

9. Do one of the following:

    • [Classic Load Balancer in EC2-Classic] On the **Instances** tab, choose **Edit Availability Zones**. On the **Add and Remove Availability Zones** page, select the Availability Zone to add.

    • [Classic Load Balancer in a VPC] On the **Instances** tab, choose **Edit Availability Zones**. On the **Add and Remove Subnets** page, for **Available subnets**, choose the add icon (+) for the subnet to add. The subnet is moved under **Selected subnets**.

    • [Application Load Balancer] On the **Description** tab, for **Availability Zones**, choose **Edit**. Choose the add icon (+) for one of the subnets for the Availability Zone to add. The subnet is moved under **Selected subnets**.

10. Choose **Save**.

Add an Availability Zone Using the AWS CLI

The commands that you'll use depend on whether your load balancer is a Classic Load Balancer in a VPC, a Classic Load Balancer in EC2-Classic, or an Application Load Balancer.

For an Auto Scaling group with a Classic Load Balancer in a VPC

1. Add a subnet to the Auto Scaling group using the following update-auto-scaling-group command:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg
 --vpc-zone-identifier subnet-41767929 subnet-cb663da2 --min-size 2
```

2. Verify that the instances in the new subnet are ready to accept traffic from the load balancer using the following describe-auto-scaling-groups command:

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

3. Enable the new subnet for your Classic Load Balancer using the following attach-load-balancer-to-subnets command:

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb --subnets subnet-41767929
```

For an Auto Scaling group with a Classic Load Balancer in EC2-Classic

1. Add an Availability Zone to the Auto Scaling group using the following update-auto-scaling-group command:

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

2. Verify that the instances in the new Availability Zone are ready to accept traffic from the load balancer using the following describe-auto-scaling-groups command:

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

3. Enable the new Availability Zone for your Classic Load Balancer using the following enable-availability-zones-for-load-balancer command:

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

For an Auto Scaling group with an Application Load Balancer

1. Add a subnet to the Auto Scaling group using the following update-auto-scaling-group command:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --vpc-zone-identifier subnet-41767929 subnet-cb663da2 --min-size 2
```

2. Verify that the instances in the new subnet are ready to accept traffic from the load balancer using the following describe-auto-scaling-groups command:

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

3. Enable the new subnet for your Application Load Balancer using the following set-subnets command:

```
aws elbv2 set-subnets --load-balancer-arn my-lb-arn --subnets subnet-41767929 subnet-cb663da2
```

Merging Your Auto Scaling Groups into a Single Multi-Zone Group

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

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

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

Merge Zones Using the AWS CLI

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

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

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

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

2. Use the following set-desired-capacity command to increase the size of my-group-a:

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

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

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

4. Use the following update-auto-scaling-group command to remove the instances from my-group-c:

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

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

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

    The following is example output:

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

```
            "AvailabilityZones": [
                "us-west-2c"
            ],
            "HealthCheckType": "EC2"
        }
    ]
}
```

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

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

Deleting Your Auto Scaling Infrastructure

To completely delete your Auto Scaling infrastructure, complete the following tasks.

Tasks

Delete Your Auto Scaling Group

When you delete an Auto Scaling group, its desired, minimum, and maximum values are set to 0. As a result, the Auto Scaling instances are terminated. Alternatively, you can terminate or detach the instances before you delete the Auto Scaling group.

To delete your Auto Scaling group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. On the Auto Scaling groups page, select your Auto Scaling group. and choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

To delete your Auto Scaling group using the AWS CLI

Use the following delete-auto-scaling-group command to delete the Auto Scaling group:

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

(Optional) Delete the Launch Configuration

Note that you can skip this step if you want to keep the launch configuration for future use.

To delete the launch configuration using the console

1. On the navigation pane, under **Auto Scaling**, choose **Launch Configurations**.
2. On the **Launch Configurations** page, select your launch configuration and choose **Actions**, **Delete launch configuration**.

3. When prompted for confirmation, choose **Yes, Delete**.

To delete the launch configuration using the AWS CLI

Use the following delete-launch-configuration command:

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-lc
```

(Optional) Delete the Load Balancer

Note that you can skip this step if your Auto Scaling group is not registered with an Elastic Load Balancing load balancer or you want to keep the load balancer for future use.

To delete your load balancer

1. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
2. Select the load balancer and choose **Actions**, **Delete**.
3. When prompted for confirmation, choose **Yes, Delete**.

To delete the load balancer associated with the Auto Scaling group using the AWS CLI

Use the following delete-load-balancer command:

```
aws elb delete-load-balancer my-load-balancer
```

(Optional) Delete CloudWatch Alarms

Note that you can skip this step if your Auto Scaling group is not associated with any CloudWatch alarms or you want to keep the alarms for future use.

To delete the CloudWatch alarms using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. On the navigation pane, choose **Alarms**.
3. Select the alarms and choose **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

To delete the CloudWatch alarms using the AWS CLI

Use the delete-alarms command. For example, use the following command to delete the AddCapacity and RemoveCapacity alarms:

```
aws cloudwatch delete-alarms --alarm-name AddCapacity RemoveCapacity
```

Scaling the Size of Your Auto Scaling Group

Scaling is the ability to increase or decrease the compute capacity of your application. Scaling starts with an event, or scaling action, which instructs Auto Scaling to either launch or terminate EC2 instances.

Auto Scaling provides a number of ways to adjust scaling to best meet the needs of your applications. As a result, it's important that you have a good understanding of your application. You should keep the following considerations in mind:

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

Contents

Scaling Plans

Auto Scaling provides several ways for you to scale your Auto Scaling group.

Maintain current instance levels at all times

You can configure your Auto Scaling group to maintain a minimum or specified number of running instances at all times. To maintain the current instance levels, Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one. For information about configuring your Auto Scaling group to maintain the current instance levels, see Maintaining the Number of Instances in Your Auto Scaling Group (p. 58).

Manual scaling

Manual scaling is the most basic way to scale your resources. You only need to specify the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Auto Scaling manages the process of creating or terminating instances to maintain the updated capacity. For more information, see Manual Scaling (p. 59).

Scale based on a schedule

Sometimes you know exactly when you will need to increase or decrease the number of instances in your group, simply because that need arises on a predictable schedule. Scaling by schedule means that scaling actions are performed automatically as a function of time and date. For more information, see Scheduled Scaling (p. 67).

Scale based on demand

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

Note that you should have two policies, one for scaling in (terminating instances) and one for scaling out (launching instances), for each event to monitor. For example, if you want to scale out when the network bandwidth reaches a certain level, create a policy specifying that Auto Scaling should start a certain number of instances to help with your traffic. But you may also want an accompanying policy to scale in by a certain number when the network bandwidth level goes back down. For more information, see Dynamic Scaling (p. 70).

Multiple Scaling Policies

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

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

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

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

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

Maintaining the Number of Instances in Your Auto Scaling Group

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

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

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

Determining Instance Health

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

If you have associated your Auto Scaling group with a load balancer or a target group and have chosen to use the ELB health checks, Auto Scaling determines the health status of the instances by checking both the instance status checks and the ELB health checks. Auto Scaling marks an instance as unhealthy if the instance is in a state other than `running`, the system status is `impaired`, or Elastic Load Balancing reports that the instance failed the health checks.

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

Replacing Unhealthy Instances

After an instance has been marked unhealthy as a result of an Amazon EC2 or Elastic Load Balancing health check, it is almost immediately scheduled for replacement. It never automatically recovers

its health. You can intervene manually by calling the SetInstanceHealth action (or the `as-set-instance-health` command) to set the instance's health status back to healthy, but you will get an error if the instance is already terminating. Because the interval between marking an instance unhealthy and its actual termination is so small, attempting to set an instance's health status back to healthy with the `SetInstanceHealth` action (or, `as-set-instance-health` command) is probably useful only for a suspended group. For more information, see Suspending and Resuming Auto Scaling Processes (p. 99).

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

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

Manual Scaling

At any time, you can change the size of an existing Auto Scaling group by updating the desired capacity of the Auto Scaling group, or by updating the instances that are attached to the Auto Scaling group.

Contents

Change the Size of Your Auto Scaling Group Using the Console

When you change the size of your Auto Scaling group, Auto Scaling manages the process of launching or terminating instances to maintain the new group size.

The following example assumes that you've created an Auto Scaling group with a minimum size of 1 and a maximum size of 5. Therefore, the group currently has one running instance.

To change the size of your Auto Scaling group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Details** tab, choose **Edit**.
5. For **Desired**, increase the desired capacity by one. For example, if the current value is `1`, type `2`.

    The desired capacity must be less than or equal to the maximum size of the group. Therefore, you must update **Max** if your new value for **Desired** is greater than **Max**.

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

Now, verify that your Auto Scaling group has launched one additional instance.

To verify that the size of your Auto Scaling group has changed

1. On the **Activity History** tab, the **Status** column shows the current status of your instance. You can use the refresh button until you see the status of your instance change to **Successful**, indicating that your Auto Scaling group has successfully launched a new instance.
2. On the **Instances** tab, the **Lifecycle** column shows the state of your instances. It takes a short time for an instance to launch. After the instance starts, its state changes to `InService`. You can see that your Auto Scaling group has launched `1` new instance, and it is in the `InService` state.

Change the Size of Your Auto Scaling Group Using the AWS CLI

When you change the size of your Auto Scaling group, Auto Scaling manages the process of launching or terminating instances to maintain the new group size.

The following example assumes that you've created an Auto Scaling group with a minimum size of 1 and a maximum size of 5. Therefore, the group currently has one running instance.

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

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg --
desired-capacity 2
```

By default, the command does not wait for the cooldown period specified for the group to complete. You can override the default behavior and wait for the cooldown period to complete by specifying the `--honor-cooldown` option as shown in the following example. For more information, see Auto Scaling Cooldowns (p. 81).

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg --
desired-capacity 2 --honor-cooldown
```

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

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

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

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "HealthCheckGracePeriod": 300,
            "SuspendedProcesses": [],
            "DesiredCapacity": 2,
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
```

```
            {
                "InstanceId": "i-33388a3f",
                "AvailabilityZone": "us-west-2a",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc"
            }
        ],
        "MaxSize": 5,
        "VPCZoneIdentifier": "subnet-e4f33493",
        "TerminationPolicies": [
            "Default"
        ],
        "LaunchConfigurationName": "my-lc",
        "CreatedTime": "2014-12-12T23:30:42.611Z",
        "AvailabilityZones": [
            "us-west-2a"
        ],
        "HealthCheckType": "EC2"
        }
    ]
}
```

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

Attach EC2 Instances to Your Auto Scaling Group

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

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

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

When you attach instances, Auto Scaling increases the desired capacity of the group by the number of instances being attached. If the number of instances being attached plus the desired capacity exceeds the maximum size of the group, the request fails.

If you attach an instance to an Auto Scaling group that has an attached load balancer, the instance is registered with the load balancer. If you attach an instance to an Auto Scaling group that has an attached target group, the instance is registered with the target group.

Contents

Note that the examples use an Auto Scaling group with the following configuration:

- Auto Scaling group name = `my-asg`
- Minimum size = `1`
- Maximum size = `5`
- Desired capacity = `2`
- Availability Zone = `us-west-2a`

Attaching an Instance Using the AWS Management Console

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

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

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Instances**.
3. Select the instance.
4. Choose **Actions**, **Instance Settings**, **Attach to Auto Scaling Group**.
5. On the **Attach to Auto Scaling Group** page, select **a new Auto Scaling group**, type a name for the group, and then choose **Attach**.

   The new Auto Scaling group is created using a new launch configuration with the same name that you specified for the Auto Scaling group. The launch configuration gets its settings (for example, security group and IAM role) from the instance that you attached. The Auto Scaling group gets settings (for example, Availability Zone and subnet) from the instance that you attached, and has a desired capacity and maximum size of `1`.
6. (Optional) To edit the settings for the Auto Scaling group, on the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**. Select the new Auto Scaling group, choose **Edit**, change the settings as needed, and then choose **Save**.

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

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. (Optional) On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**. Select the Auto Scaling group and verify that the maximum size of the Auto Scaling group is large enough that you can add another instance. Otherwise, choose **Edit**, increase the maximum size, and then choose **Save**.
3. On the navigation pane, choose **Instances**.
4. Select the instance.
5. Choose **Actions**, **Instance Settings**, **Attach to Auto Scaling Group**.
6. On the **Attach to Auto Scaling Group** page, select **an existing Auto Scaling group**, select the instance, and then choose **Attach**.
7. If the instance doesn't meet the criteria (for example, if it's not in the same Availability Zone as the Auto Scaling group), you'll get an error message with the details. Choose **Close** and try again with an instance that meets the criteria.

Attaching an Instance Using the AWS CLI

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

1. Describe a specific Auto Scaling group using the following describe-auto-scaling-groups command:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-
names my-asg
```

The following example response shows that the desired capacity is 2 and the group has 2 running instances:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "HealthCheckGracePeriod": 300,
            "SuspendedProcesses": [],
            "DesiredCapacity": 2,
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "InstanceId": "i-a5e87793",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "my-lc"
                },
                {
                    "InstanceId": "i-a4e87792",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "my-lc"
                }
            ],
            "MaxSize": 5,
            "VPCZoneIdentifier": "subnet-e4f33493",
            "TerminationPolicies": [
                "Default"
            ],
            "LaunchConfigurationName": "my-lc",
            "CreatedTime": "2014-12-12T23:30:42.611Z",
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "HealthCheckType": "EC2"
        }
    ]
}
```

2. Attach an instance to the Auto Scaling group using the following attach-instances command:

```
aws autoscaling attach-instances --instance-ids i-a8e09d9c --auto-scaling-
group-name my-asg
```

3. To verify that the instance is attached, use the following `describe-auto-scaling-groups` command:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-
names my-asg
```

The following example response shows that the desired capacity has increased by 1 to 3, and that there is a new instance, `i-a8e09d9c`:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "HealthCheckGracePeriod": 300,
            "SuspendedProcesses": [],
            "DesiredCapacity": 3,
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "InstanceId": "i-a8e09d9c",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "my-lc"
                },
                {
                    "InstanceId": "i-a5e87793",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "my-lc"
                },
                {
                    "InstanceId": "i-a4e87792",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "LaunchConfigurationName": "my-lc"
                }
            ],
            "MaxSize": 5,
            "VPCZoneIdentifier": "subnet-e4f33493",
            "TerminationPolicies": [
                "Default"
            ],
            "LaunchConfigurationName": "my-lc",
            "CreatedTime": "2014-12-12T23:30:42.611Z",
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "HealthCheckType": "EC2"
        }
```

```
        ]
}
```

Detach EC2 Instances From Your Auto Scaling Group

You can remove an instance from an Auto Scaling group. After the instances are detached, you can manage them independently from the rest of the Auto Scaling group. By detaching an instance, you can:

- Move an instance out of one Auto Scaling group and attach it to a different one. For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 61).
- Test an Auto Scaling group by creating it using existing instances running your application, and then detach these instances from the Auto Scaling group when your tests are complete.

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

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

Contents

- Detaching Instances Using the AWS Management Console (p. 65)
- Detaching Instances Using the AWS CLI (p. 66)

Note that the examples use an Auto Scaling group with the following configuration:

- Auto Scaling group name = `my-asg`
- Minimum size = `1`
- Maximum size = `5`
- Desired capacity = `4`
- Availability Zone = `us-west-2a`

Detaching Instances Using the AWS Management Console

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

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

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Instances** tab, select the instance and choose **Actions**, **Detach**.
5. On the **Detach Instance** page, select the checkbox if you want Auto Scaling to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Choose **Detach Instance**.

Detaching Instances Using the AWS CLI

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

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

1. List the current instances using the following describe-auto-scaling-instances command:

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

    The following example response shows that the group has 4 running instances:

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

2. Detach an instance and decrement the desired capacity using the following detach-instances command:

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

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

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

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

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

Scheduled Scaling

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

To configure your Auto Scaling group to scale based on a schedule, you create a scheduled action, which tells Auto Scaling to perform a scaling action at specified times. To create a scheduled scaling action, you specify the start time when you want the scaling action to take effect, and the new minimum, maximum, and desired sizes for the scaling action. At the specified time, Auto Scaling updates the group with the values for minimum, maximum, and desired size specified by the scaling action.

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

Contents

-
-

Considerations for Scheduled Actions

When you create a scheduled action, keep the following 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 can take longer to execute.
- You can create a maximum of 125 scheduled actions per Auto Scaling group.
- A scheduled action must have a unique time value. If you attempt to schedule an activity at a time when another scaling activity is already scheduled, the call is rejected with an error message noting the conflict.
- Cooldown periods are not supported.

Create a Scheduled Action Using the Console

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

To create a scheduled action

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Scheduled Actions** tab, choose **Create Scheduled Action**.
5. On the **Create Scheduled Action** page, do the following:

   a. Specify the size of the group using at least one of **Min**, **Max**, and **Desired Capacity**.

   b. Choose an option for **Recurrence**. If you choose **Once**, Auto Scaling performs the action at the specified time. If you select **Cron**, type a Cron expression that specifies when Auto Scaling performs the action, in UTC. If you select an option that begins with **Every**, the Cron expression is created for you.

   c. If you chose **Once** for **Recurrence**, specify the time for the action in **Start Time**.

   d. If you specified a recurring schedule, you can specify values for **Start Time** and **End Time**. If you specify a start time, Auto Scaling performs the action at this time, and then performs the action based on the recurring schedule. If you specify an end time, Auto Scaling does not perform the action after this time.

   e. Choose **Create**.

Update a Scheduled Action

If your requirements change, you can update a scheduled action.

To update a scheduled action

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3. Select your Auto Scaling group.

4. On the **Scheduled Actions** tab, select the scheduled action.

5. Choose **Actions**, **Edit**.

6. On the **Edit Scheduled Action** page, do the following:

    a. Update the size of the group as needed using **Min**, **Max**, or **Desired Capacity**.

    b. Update the specified recurrence as needed.

    c. Update the start and end time as needed.

    d. Choose **Save**.

Create or Update a Scheduled Action Using the AWS CLI

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

To schedule scaling for one time only

To increase the number of running instances in your Auto Scaling group at a specific time, in "YYYY-MM-DDThh:mm:ssZ" format in UTC, use the following put-scheduled-update-group-action command:

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

To decrease the number of running instances in your Auto Scaling group at a specific time, in "YYYY-MM-DDThh:mm:ssZ" format in UTC, use the following put-scheduled-update-group-action command:

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

To schedule scaling on a recurring schedule

You can specify a recurrence schedule, in UTC, using the Cron format. For more information, see the Cron Wikipedia entry.

Use the following put-scheduled-update-group-action command to create a scheduled action that runs at 00:30 hours on the first of January, June, and December each year:

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

Delete a Scheduled Action

When you are finished with a scheduled action, you can delete it.

To delete a scheduled action using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3. Select your Auto Scaling group.

4. On the **Scheduled Actions** tab, select the scheduled action.

5. Choose **Actions**, **Delete**.

6. When prompted for confirmation, choose **Yes, Delete**.

To delete a scheduled action using the AWS CLI

Use the following delete-scheduled-action command:

```
aws autoscaling delete-scheduled-action --scheduled-action-name ScaleUp
```

Dynamic Scaling

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

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

To set up dynamic scaling, you must create alarms and scaling policies and associate them with your Auto Scaling group. We recommend that you create two policies for each scaling change that you want to perform: one policy to scale out and another policy to scale in. After the alarm sends a message to Auto Scaling, Auto Scaling executes the associated policy to scale your group in (by terminating instances) or out (by launching instances). The process is as follows:

1. Amazon CloudWatch monitors the specified metrics for all the instances in the Auto Scaling group.

2. As demand grows or shrinks, the change is reflected in the metrics.

3. When the change in the metrics breaches the threshold of the CloudWatch alarm, the alarm performs an action. Depending on the breach, the action is a message sent to either the scale-in policy or the scale-out policy.

4. After the Auto Scaling policy receives the message, Auto Scaling performs the scaling activity for the Auto Scaling group.

5. This process continues until you delete either the scaling policies or the Auto Scaling group.

Contents

-

Scaling Adjustment Types

When a scaling policy is executed, it changes the current capacity of your Auto Scaling group using the scaling adjustment specified in the policy. A scaling adjustment can't change the capacity of the group above the maximum group size or below the minimum group size.

Auto Scaling supports the following adjustment types:

- **ChangeInCapacity**—Increase or decrease the current capacity of the group by the specified number of instances. A positive value increases the capacity and a negative adjustment value decreases the capacity.

  Example: If the current capacity of the group is 3 instances and the adjustment is 5, then when this policy is performed, Auto Scaling adds 5 instances to the group for a total of 8 instances.
- **ExactCapacity**—Change the current capacity of the group to the specified number of instances. Note that you must specify a positive value with this adjustment type.

  Example: If the current capacity of the group is 3 instances and the adjustment is 5, then when this policy is performed, Auto Scaling changes the capacity to 5 instances.
- **PercentChangeInCapacity**—Increment or decrement the current capacity of the group by the specified percentage. A positive value increases the capacity and a negative value decreases the capacity. If the resulting value is not an integer, Auto Scaling rounds it as follows:
  - Values greater than 1 are rounded down. For example, `12.7` is rounded to `12`.
  - Values between 0 and 1 are rounded to 1. For example, `.67` is rounded to `1`.
  - Values between 0 and -1 are rounded to -1. For example, `-.58` is rounded to `-1`.
  - Values less than -1 are rounded up. For example, `-6.67` is rounded to `-6`.

  Example: If the current capacity is 10 instances and the adjustment is 10 percent, then when this policy is performed, Auto Scaling adds 1 instance to the group for a total of 11 instances.

Scaling Policy Types

When you create a scaling policy, you must specify its policy type. The policy type determines how the scaling action is performed. Auto Scaling supports the following policy types:

- **Simple scaling**—Increase or decrease the current capacity of the group based on a single scaling adjustment.
- **Step scaling**—Increase or decrease the current capacity of the group based on a set of scaling adjustments, known as *step adjustments*, that vary based on the size of the alarm breach.

Simple Scaling Policies

After a scaling activity is started, the policy must wait for the scaling activity or health check replacement to complete and the cooldown period to expire before it can respond to additional alarms. Cooldown periods help to prevent Auto Scaling from initiating additional scaling activities before the effects of previous activities are visible. You can use the default cooldown period associated with your Auto Scaling group, or you can override the default by specifying a cooldown period for your policy. For more information, see Auto Scaling Cooldowns (p. 81).

Note that Auto Scaling originally supported only this type of scaling policy. If you created your scaling policy before policy types were introduced, your policy is treated as a simple scaling policy.

Step Scaling Policies

After a scaling activity is started, the policy continues to respond to additional alarms, even while a scaling activity or health check replacement is in progress. Therefore, all alarms that are breached are evaluated by Auto Scaling as it receives the alarm messages. If you are creating a policy to scale out, you can specify the estimated warm-up time that it will take for a newly launched instance to be ready to contribute to the aggregated metrics. For more information, see Instance Warmup (p. 73).

> **Note**
> Cooldown periods are not supported for step scaling policies. Therefore, you can't specify a cooldown period for these policies and the default cooldown period for the group doesn't apply.

We recommend that you use step scaling policies even if you have a single step adjustment, because we continuously evaluate alarms and do not lock the group during scaling activities or health check replacements.

Step Adjustments

When you create a step scaling policy, you add one or more step adjustments, which enables you to scale based on the size of the alarm breach. Each step adjustment specifies a lower bound for the metric value, an upper bound for the metric value, and the amount by which to scale, based on the scaling adjustment type.

There are a few rules for the step adjustments for your policy:

- The ranges of your step adjustments can't overlap or have a gap.
- At most one step adjustment can have a null lower bound (negative infinity). If one step adjustment has a negative lower bound, then there must be a step adjustment with a null lower bound.
- At most one step adjustment can have a null upper bound (positive infinity). If one step adjustment has a positive upper bound, then there must be a step adjustment with a null upper bound.
- The upper and lower bound can't be null in the same step adjustment.
- If the metric value is above the breach threshold, the lower bound is inclusive and the upper bound is exclusive. If the metric value is below the breach threshold, the lower bound is exclusive and the upper bound is inclusive.

If you are using the API or the CLI, you specify the upper and lower bounds relative to the value of the aggregated metric. If you are using the AWS Management Console, you specify the upper and lower bounds as absolute values.

Auto Scaling applies the aggregation type to the metric data points from all instances and compares the aggregated metric value against the upper and lower bounds defined by the step adjustments to determine which step adjustment to perform. For example, suppose that you have an alarm with a breach threshold of 50 and a scaling adjustment type of `PercentChangeInCapacity`. You also have scale out and scale in policies with the following step adjustments:

Scale out policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
0	10	0	50 <= *value* < 60
10	20	10	60 <= *value* < 70
20	null	30	70 <= *value* < +infinity

Scale in policy			
Lower bound	**Upper bound**	**Adjustment**	**Metric value**
-10	0	0	40 < *value* <= 50
-20	-10	-10	30 < *value* <= 40
null	-20	-30	-infinity < *value* <= 30

Your group has both a current capacity and a desired capacity of 10 instances. The group maintains its current and desired capacity while the aggregated metric value is greater than 40 and less than 60.

If the metric value gets to 60, Auto Scaling increases the desired capacity of the group by 1 instance, to 11 instances, based on the second step adjustment of the scale-out policy (add 10 percent of 10 instances). After the new instance is running and its specified warm-up time has expired, Auto Scaling increases the current capacity of the group to 11 instances. If the metric value rises to 70 even after this increase in capacity, Auto Scaling increases the desired capacity of the group by another 3 instances, to 14 instances, based on the third step adjustment of the scale-out policy (add 30 percent of 11 instances, 3.3 instances, rounded down to 3 instances).

If the metric value gets to 40, Auto Scaling decreases the desired capacity of the group by 1 instance, to 13 instances, based on the second step adjustment of the scale-in policy (remove 10 percent of 14 instances, 1.4 instances, rounded down to 1 instance). If the metric value falls to 30 even after this decrease in capacity, Auto Scaling decreases the desired capacity of the group by another 3 instances, to 10 instances, based on the third step adjustment of the scale-in policy (remove 30 percent of 13 instances, 3.9 instances, rounded down to 3 instances).

Instance Warmup

With step scaling policies, you can specify the number of seconds that it takes for a newly launched instance to warm up. Until its specified warm-up time has expired, an instance is not counted toward the aggregated metrics of the Auto Scaling group.

While scaling out, Auto Scaling does not consider instances that are warming up as part of the current capacity of the group. Therefore, multiple alarm breaches that fall in the range of the same step adjustment result in a single scaling activity. This ensures that we don't add more instances than you need. Using the example in the previous section, suppose that the metric gets to 60, and then it gets to 62 while the new instance is still warming up. The current capacity is still 10 instances, so Auto Scaling should add 1 instance (10 percent of 10 instances), but the desired capacity of the group is already 11 instances, so Auto Scaling does not increase the desired capacity further. However, if the metric gets to 70 while the new instance is still warming up, Auto Scaling should add 3 instances (30 percent of 10 instances), but the desired capacity of the group is already 11, so Auto Scaling adds only 2 instances, for a new desired capacity of 13 instances.

While scaling in, Auto Scaling considers instances that are terminating as part of the current capacity of the group. Therefore, we won't remove more instances from the Auto Scaling group than necessary.

Note that a scale in activity can't start while a scale out activity is in progress.

Scaling Based on Metrics

You can create a scaling policy that uses CloudWatch alarms to determine when your Auto Scaling group should scale out or scale in. Each CloudWatch alarm watches a single metric and sends messages to Auto Scaling when the metric breaches a threshold that you specify in your policy. You can use alarms to monitor any of the metrics that the services in AWS that you're using send to CloudWatch, or you can create and monitor your own custom metrics.

When you create a CloudWatch alarm, you can specify an Amazon SNS topic to send an email notification to when the alarm changes state. For more information, see Create Amazon CloudWatch Alarms (p. 110).

Contents

Create an Auto Scaling Group with Scaling Policies

Use the console to create an Auto Scaling group with two scaling policies: a scale out policy that increases the capacity of the group by 30 percent, and a scale in policy that decreases the capacity of the group to two instances.

To create an Auto Scaling group with scaling based on metrics

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3. Choose **Create Auto Scaling group**.

4. On the **Create Auto Scaling Group** page, do one of the following:

    * Select **Create an Auto Scaling group from an existing launch configuration**, select an existing launch configuration, and then choose **Next Step**.

    * If you don't have a launch configuration that you'd like to use, choose **Create a new launch configuration** and follow the directions. For more information, see Creating a Launch Configuration (p. 21).

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

    a. For **Group name**, type a name for your Auto Scaling group.

    b. For **Group size**, type the desired capacity for your Auto Scaling group.

    c. If the launch configuration specifies instances that require a VPC, such as T2 instances, you must select a VPC from **Network**. Otherwise, if your AWS account supports EC2-Classic and the instances don't require a VPC, you can select either `Launch info EC2-Classic` or a VPC.

    d. If you selected a VPC in the previous step, select one or more subnets from **Subnet**. If you selected EC2-Classic in the previous step, select one or more Availability Zones from **Availability Zone(s)**.

    e. Choose **Next: Configure scaling policies**.

6. On the **Configure scaling policies** page, do the following:

    a. Select **Use scaling policies to adjust the capacity of this group**.

    b. Specify the minimum and maximum size for your Auto Scaling group using the row that begins with **Scale between**. For example, if your group is already at its maximum size, you need to specify a new maximum in order to scale out.

    Scale between 1 and 5 instances. These will be the minimum and maximum size of your group.

    c. Specify your scale out policy under **Increase Group Size**. You can optionally specify a name for the policy, then choose **Add new alarm**.

    d. On the **Create Alarm** page, choose **create topic**. For **Send a notification to**, type a name for the SNS topic. For **With these recipients**, type one or more email addresses to receive notification. If you want, you can replace the default name for your alarm with a custom name. Next, specify the metric and the criteria for the policy. For example, you can leave the default

settings for **Whenever** (Average of CPU Utilization). For **Is**, choose `>=` and type `80` percent. For **For at least**, type `1` consecutive period of `5 Minutes`. Choose **Create Alarm**.

e. For **Take the action**, choose `Add`, type `30` in the next field, and then choose `percent of group`. By default, the lower bound for this step adjustment is the alarm threshold and the upper bound is null (positive infinity). To add another step adjustment, choose **Add step**.

(Optional) We recommend that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**. For more information, see Scaling Policy Types (p. 71).

f. Specify your scale in policy under **Decrease Group Size**. You can optionally specify a name for the policy, then choose **Add new alarm**.

g. On the **Create Alarm** page, you can select the same notification that you created for the scale out policy or create a new one for the scale in policy. If you want, you can replace the default name for your alarm with a custom name. Keep the default settings for **Whenever** (Average of CPU Utilization). For **Is**, choose `<=` and type `40` percent. For **For at least**, type `1` consecutive period of `5 Minutes`. Choose **Create Alarm**.

h. For **Take the action**, choose `Remove`, type `2` in the next field, and then choose `instances`. By default, the upper bound for this step adjustment is the alarm threshold and the lower bound is null (negative infinity). To add another step adjustment, choose **Add step**.

(Optional) We recommend that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**. For more information, see Scaling Policy Types (p. 71).

Decrease Group Size

Name: DecreaseCapacity

Execute policy when: DecreaseCapacityAlarm Edit Remove
breaches the alarm threshold: CPUUtilization <= 40 for 300 seconds
for the metric dimensions AutoScalingGroupName = my-asg

Take the action: Remove ▾ 2 instances ▾ when 40 >= CPUUtilization > -infinity
Add step ⓘ

Create a simple scaling policy ⓘ

    i.   Choose **Review**.

    j.   On the **Review** page, choose **Create Auto Scaling group**.

7.   Use the following steps to verify the scaling policies for your Auto Scaling group.

    a.   The **Auto Scaling Group creation status** page confirms that your Auto Scaling group was successfully created. Choose **View your Auto Scaling Groups**.

    b.   On the **Auto Scaling Groups** page, select the Auto Scaling group that you just created.

    c.   On the **Activity History** tab, the **Status** column shows whether your Auto Scaling group has successfully launched instances.

    d.   On the **Instances** tab, the **Lifecycle** column contains the state of your instances. It takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to `InService`.

        The **Health Status** column shows the result of the EC2 instance health check on your instance.

    e.   On the **Scaling Policies** tab, you can see the policies that you created for the Auto Scaling group.

Add a Scaling Policy to an Auto Scaling Group

Use the console to add a scaling policy to an existing Auto Scaling group.

To update an Auto Scaling group with scaling based on metrics

1.   Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2.   On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.

3.   Select the Auto Scaling group.

4.   On the **Scaling Policies** tab, choose **Add policy**.

5.   For **Name**, type a name for the policy, and then choose **Create new alarm**.

6.   On the **Create Alarm** page, choose **create topic**. For **Send a notification to**, type a name for the SNS topic. For **With these recipients**, type one or more email addresses to receive notification. If you want, you can replace the default name for your alarm with a custom name. Next, specify the metric and the criteria for the alarm, using **Whenever**, **Is**, and **For at least**. Choose **Create Alarm**.

7.   Specify the scaling activity for the policy using **Take the action**. By default, the lower bound for this step adjustment is the alarm threshold and the upper bound is null (positive infinity). To add another step adjustment, choose **Add step**.

    (Optional) We recommend that you use the default to create both scaling policies with steps. If you need to create simple scaling policies, choose **Create a simple scaling policy**. For more information, see Scaling Policy Types (p. 71).

8.   Choose **Create**.

Configure Scaling Policies Using the AWS CLI

Use the AWS CLI as follows to configure scaling policies for your Auto Scaling group.

Tasks

Step 1: Create an Auto Scaling Group

Use the following create-auto-scaling-group command to create an Auto Scaling group named `my-asg` using the launch configuration `my-lc`. If you don't have a launch configuration that you'd like to use, you can create one. For more information, see create-launch-configuration.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg
 --launch-configuration-name my-lc --max-size 5 --min-size 1 --availability-
zones "us-west-2c"
```

Step 2: Create Scaling Policies

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

Example: my-scaleout-policy

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

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

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

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-
a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-scaleout-policy
}
```

Example: my-scalein-policy

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

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

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

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-
aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-scalein-policy
}
```

Step 3: Create CloudWatch Alarms

In step 2, you created scaling policies that provided instructions to the Auto Scaling group about how to scale out and scale in when the conditions that you specify change. In this step, you create alarms by identifying the metrics to watch, defining the conditions for scaling, and then associating the alarms with the scaling policies.

Example: AddCapacity

Use the following CloudWatch put-metric-alarm command to create an alarm that increases the size of the Auto Scaling group when the value of the specified metric breaches 80. For example, you can add capacity when the average CPU usage of all the instances (CPUUtilization) increases to 80 percent. To use your own custom metric, specify its name in --metric-name and its namespace in --namespace.

```
aws cloudwatch put-metric-alarm --alarm-name AddCapacity --metric-
name CPUUtilization --namespace AWS/EC2
--statistic Average --period 120 --threshold 80 --comparison-operator
 GreaterThanOrEqualToThreshold
--dimensions "Name=AutoScalingGroupName,Value=my-asg" --evaluation-periods 2
 --alarm-actions PolicyARN
```

Example: RemoveCapacity

Use the following CloudWatch put-metric-alarm command to create an alarm that decreases the size of the Auto Scaling group when the value of the specified metric breaches 40. For example, you can remove capacity when the average CPU usage of all the instances (CPUUtilization) decreases to 40 percent. To use your own custom metric, specify its name in --metric-name and its namespace in --namespace.

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

Scaling Based on Amazon SQS

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

For example, suppose that you have a web app that receives orders from customers. The app runs on EC2 instances in an Auto Scaling group that is configured to handle a typical amount of orders. The

app places the orders in an Amazon SQS queue until they are picked up for processing, processes the orders, and then sends the processed orders back to the customer. The following diagram illustrates the architecture of this example.
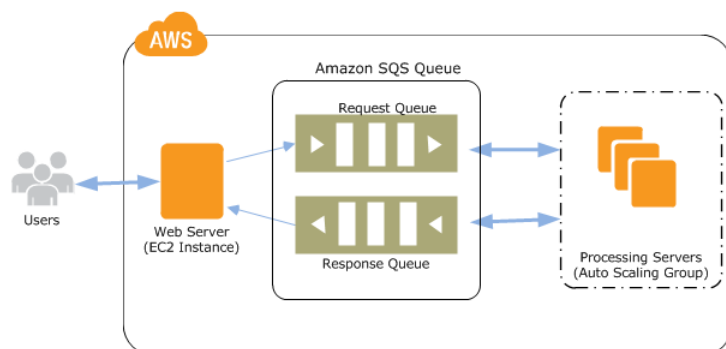
This architecture works well if your order levels remain the same at all times. What happens if your order levels change? You would need to launch additional EC2 instances when the orders increase and terminate the extra EC2 instances when the orders decrease. If your orders increase and decrease on a predictable schedule, you can specify the time and date to perform scaling activities. For more information, see Scheduled Scaling (p. 67). Otherwise, you can scale based on criteria, such as the number of messages in your SQS queue. For more information, see Dynamic Scaling (p. 70).

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

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

Scaling with Amazon SQS Using the AWS CLI

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

Create the Scaling Policies

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

To create scaling policies

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

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

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

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

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

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

Create the CloudWatch Alarms

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

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

To create CloudWatch alarms

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

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

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

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

Verify Your Scaling Policies and CloudWatch Alarms

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

To verify your CloudWatch alarms

Use the following describe-alarms command:

```
aws cloudwatch describe-alarms --alarm-
names AddCapacityToProcessQueue RemoveCapacityFromProcessQueue
```

To verify your scaling policies

Use the following describe-policies command:

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

Test Your Scale Out and Scale In Policies

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

To test the scale out policy

1. Follow the steps in Getting Started with Amazon SQS to add messages to your SQS queue. Make sure that you have at least three messages in the queue.

   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.

2. Use the following describe-auto-scaling-groups command to verify that the group has launched an instance:

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

To test the scale in policy

1. Follow the steps in Getting Started with Amazon SQS to remove messages from the SQS queue. Make sure that you have no more than one message in the queue.

   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.

2. Use the following describe-auto-scaling-groups command to verify that the group has terminated an instance:

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

Auto Scaling Cooldowns

The Auto Scaling cooldown period is a configurable setting for your Auto Scaling group that helps to ensure that Auto Scaling doesn't launch or terminate additional instances before the previous scaling activity takes effect. After the Auto Scaling group dynamically scales using a simple scaling policy, Auto Scaling waits for the cooldown period to complete before resuming scaling activities. When you manually scale your Auto Scaling group, the default is not to wait for the cooldown period, but you can override the default and honor the cooldown period. Note that if an instance becomes unhealthy, Auto Scaling does not wait for the cooldown period to complete before replacing the unhealthy instance.

Important

Cooldown periods are not supported for step scaling policies or scheduled scaling.

Auto Scaling supports both default cooldown periods and scaling-specific cooldown periods.

Contents

Example: Auto Scaling Cooldowns

Consider the following scenario: you have a web application running in AWS. This web application consists three basic tiers: web, application, and database. To make sure that the application always has the resources that it needs to meet traffic demands, you create two Auto Scaling groups: one for your web tier and one for your application tier.
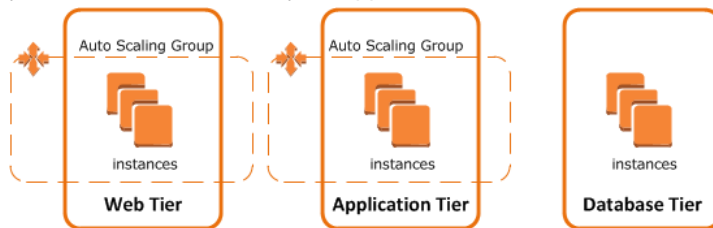
To help ensure that the Auto Scaling group for the application tier has the appropriate number of EC2 instances, create a CloudWatch alarm (p. 110) to scale out whenever the **CPUUtilization** metric for the instances exceeds 90%. When the alarm occurs, Auto Scaling launches and configures another instance.

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

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

However, with a cooldown period in place, Auto Scaling launches an instance and then suspends scaling activities due to simple scaling policies or manual scaling until the specified time elapses. (The default is 300 seconds.) This gives newly-launched instances time to start handling application traffic. After the cooldown period expires, any suspended scaling actions resume. If the CloudWatch alarm fires again, Auto Scaling launches another instance, and the cooldown period takes effect again. If, however, the additional instance was enough to bring the CPU utilization back down, then the group remains at its current size.

Default Cooldowns

The default cooldown period is applied when you create your Auto Scaling group. Its default value is 300 seconds. This cooldown period automatically applies to any dynamic scaling (p. 70) activities for simple scaling policies, and you can optionally request that it apply to your manual scaling (p. 59) activities.
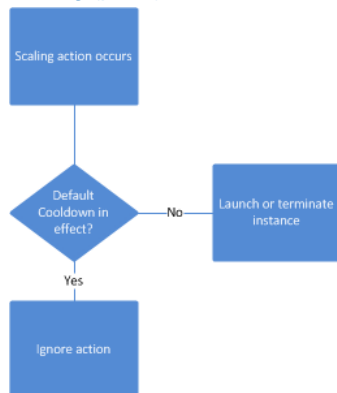
You can configure the default cooldown period when you create the Auto Scaling group, using the AWS Management Console, the create-auto-scaling-group command (AWS CLI), or the CreateAutoScalingGroup API operation.

You can change the default cooldown period whenever you need to, using the AWS Management Console, the update-auto-scaling-group command (AWS CLI), or the UpdateAutoScalingGroup API operation.

Scaling-Specific Cooldowns

In addition to specifying the default cooldown period for your Auto Scaling group, you can create cooldowns that apply to a specific simple scaling policy or manual scaling. A scaling-specific cooldown period overrides the default cooldown period.

One common use for scaling-specific cooldowns is with a scale in policy—a policy that terminates instances based on a specific criteria or metric. Because this policy terminates instances, Auto Scaling needs less time to determine whether to terminate additional instances. The default cooldown period of 300 seconds is too long—you can reduce costs by applying a scaling-specific cooldown period of 180 seconds to the scale in policy.

You can create a scaling-specific cooldown period using the AWS Management Console, the put-scaling-policy command (AWS CLI), or the PutScalingPolicy API operation.

Cooldowns and Multiple Instances

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

With multiple instances, the cooldown period (either the default cooldown or the scaling-specific cooldown) takes effect starting when the last instance launches.

Cooldowns and Lifecycle Hooks

Auto Scaling supports adding lifecycle hooks to Auto Scaling groups. These hooks enable you to control how instances launch and terminate within an Auto Scaling group; you can perform actions on the instance before it is put into service or before it is terminated.

Lifecycle hooks can affect the impact of any cooldown periods configured for the Auto Scaling group, manual scaling, or a simple scaling policy. The cooldown period does not begin until after the instance moves out of the wait state.

Cooldowns and Spot Instances

You can create Auto Scaling groups to use Spot Instances (p. 30) instead of On-Demand or Reserved Instances. The cooldown period begins when the bid for any Spot Instance is successful.

Controlling Which Instances Auto Scaling Terminates During Scale In

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

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

You can also use instance protection to prevent Auto Scaling from selecting specific instances for termination when scaling in.

Contents

Default Termination Policy

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

1. Auto Scaling determines whether there are instances in multiple Availability Zones. If so, it selects the Availability Zone with the most instances and at least one instance that is not protected from scale in. If there is more than one Availability Zone with this number of instances, Auto Scaling selects the Availability Zone with the instances that use the oldest launch configuration.
2. Auto Scaling determines which unprotected instances in the selected Availability Zone use the oldest launch configuration. If there is one such instance, it terminates it.
3. If there are multiple instances that use the oldest launch configuration, Auto Scaling determines which unprotected instances are closest to the next billing hour. (This helps you maximize the use of your EC2 instances while minimizing the number of hours you are billed for Amazon EC2 usage.) If there is one such instance, Auto Scaling terminates it.

4. If there is more than one unprotected instance closest to the next billing hour, Auto Scaling selects one of these instances at random.

The following flow diagram illustrates how the default termination policy works.
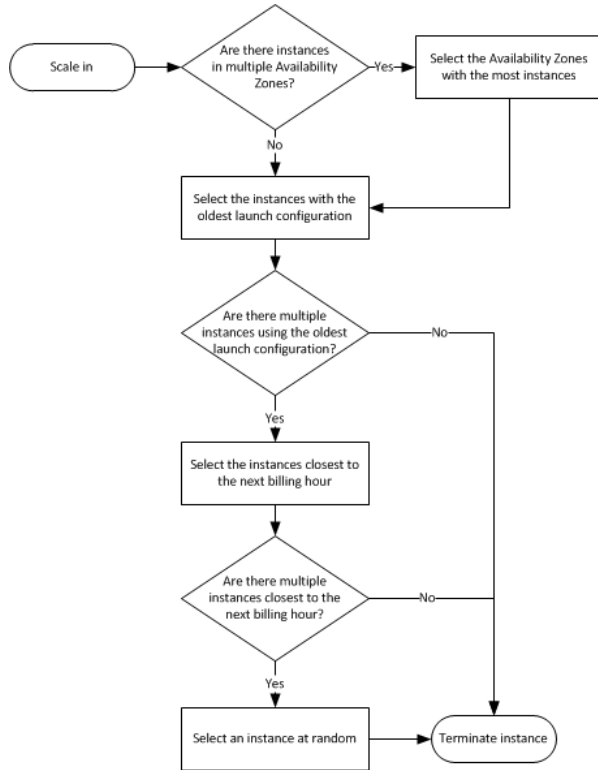
Consider an Auto Scaling group that has two Availability Zones, a desired capacity of two instances, and scaling policies that increase and decrease the number of instances by 1 when certain thresholds are met. The two instances in this group are distributed as follows.
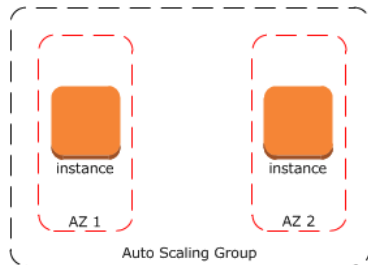
When the threshold for the scale out policy is met, the policy takes effect and Auto Scaling launches a new instance. The Auto Scaling group now has three instances, distributed as follows.
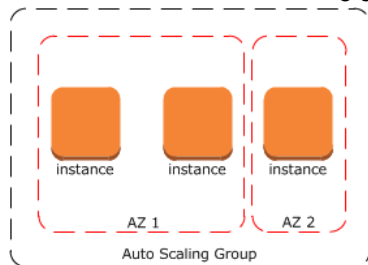
When the threshold for the scale in policy is met, the policy takes effect and Auto Scaling terminates one of the instances. If the group does not have a specific termination policy assigned to it, Auto Scaling uses the default termination policy. Auto Scaling selects the Availability Zone with two instances, and terminates the instance launched from the oldest launch configuration. If the instances were launched from the same launch configuration, then Auto Scaling selects the instance that is closest to the next billing hour and terminates it.

Customizing the Termination Policy

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

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

Auto Scaling currently supports the following custom termination policies:

- `OldestInstance`. Auto Scaling terminates the oldest instance in the group. This option is useful when you're upgrading the instances in the Auto Scaling group to a new EC2 instance type, so you can gradually replace instances of the old type with instances of the new type.
- `NewestInstance`. Auto Scaling terminates the newest instance in the group. This policy is useful when you're testing a new launch configuration but don't want to keep it in production.
- `OldestLaunchConfiguration`. Auto Scaling terminates instances that have the oldest launch configuration. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
- `ClosestToNextInstanceHour`. Auto Scaling terminates instances that are closest to the next billing hour. This policy helps you maximize the use of your instances and manage costs.
- `Default`. Auto Scaling uses its default termination policy. This policy is useful when you have more than one scaling policy associated with the group.

To customize a termination policy using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group.
4. For **Actions**, choose **Edit**.
5. On the **Details** tab, locate **Termination Policies**. Choose one or more termination policies. If you choose multiple policies, list them in the order that you would like them to apply. If you use the `Default` policy, make it the last one in the list.
6. Choose **Save**.

To customize a termination policy using the AWS CLI

Use one of the following commands:

- create-auto-scaling-group
- update-auto-scaling-group

You can use these policies individually, or combine them into a list of policies that Auto Scaling uses when terminating instances. For example, use the following command to update an Auto

Scaling group to use the `OldestLaunchConfiguration` policy first, and then to use the `ClosestToNextInstanceHour` policy:

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

If you use the `Default` termination policy, make it the last one in the list of termination policies. For example, `--termination-policies "OldestLaunchConfiguration,Default"`.

Instance Protection

To control whether Auto Scaling can terminate a particular instance when scaling in, use instance protection. You can enable the instance protection setting on an Auto Scaling group or an individual Auto Scaling instance. When Auto Scaling launches an instance, the instance inherits the instance protection setting of the Auto Scaling group. You can change the instance protection setting for an Auto Scaling group or an Auto Scaling instance at any time.

Instance protection starts when the instance state is `InService`. If you detach an instance that is protected from termination, its instance protection setting is lost. When you attach the instance to the group again, it inherits the current instance protection setting of the group.

If all instances in an Auto Scaling group are protected from termination during scale in and a scale in event occurs, Auto Scaling decrements the desired capacity. However, Auto Scaling can't terminate the required number of instances until their instance protection settings are disabled.

Instance protection does not protect Auto Scaling instances from manual termination through the Amazon EC2 console, the `terminate-instances` command, or the `TerminateInstances` operation. Instance protection does not protect an Auto Scaling instance from termination if it fails health checks and must be replaced. Also, instance protection does not protect Spot instances in an Auto Scaling group from interruption.

Tasks

- Enable Instance Protection for a Group (p. 87)
- Modify the Instance Protection Setting for a Group (p. 88)
- Modify the Instance Protection Setting for an Instance (p. 89)

Enable Instance Protection for a Group

You can enable instance protection when you create an Auto Scaling group. By default, instance protection is disabled.

To enable instance protection using the console

When you create the Auto Scaling group, on the **Configure Auto Scaling group details** page, under **Advanced Details**, select the `Protect From Scale In` option from **Instance Protection**.

To enable instance protection using the AWS CLI

Use the following create-auto-scaling-group command to enable instance protection:

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --
new-instances-protected-from-scale-in ...
```

Modify the Instance Protection Setting for a Group

You can enable or disable the instance protection setting for an Auto Scaling group.

To change the instance protection setting for a group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group.
4. On the **Details** tab, choose **Edit**.
5. For **Instance Protection**, select `Protect From Scale In`.
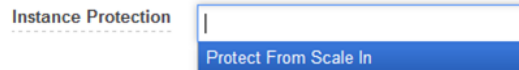
6. Choose **Save**.

To change the instance protection setting for a group using the AWS CLI

Use the following update-auto-scaling-group command to enable instance protection for the specified Auto Scaling group:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
new-instances-protected-from-scale-in
```

Use the following command to disable instance protection for the specified group:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
no-new-instances-protected-from-scale-in
```

Modify the Instance Protection Setting for an Instance

By default, an instance gets its instance protection setting from its Auto Scaling group. However, you can enable or disable instance protection for an instance at any time.

To change the instance protection setting for an instance using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. On the navigation pane, choose **Auto Scaling Groups**.

3. Select the Auto Scaling group.

4. On the **Instances** tab, select the instance.

5. To enable instance protection, choose **Actions**, **Instance Protection**, **Set Scale In Protection**. When prompted, choose **Set Scale In Protection**.

6. To disable instance protection, choose **Actions**, **Instance Protection**, **Remove Scale In Protection**. When prompted, choose **Remove Scale In Protection**.

To change the instance protection setting for an instance using the AWS CLI

Use the following set-instance-protection command to enable instance protection for the specified instance:

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-
scaling-group-name my-asg --protected-from-scale-in
```

Use the following command to disable instance protection for the specified instance:

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-
scaling-group-name my-asg --no-protected-from-scale-in
```

Auto Scaling Lifecycle Hooks

Auto Scaling *lifecycle hooks* enable you to perform custom actions as Auto Scaling launches or terminates instances. For example, you could install or configure software on newly launched instances, or download log files from an instance before it terminates.

Contents

How Lifecycle Hooks Work

After you add lifecycle hooks to your Auto Scaling group, they work as follows:

1. Auto Scaling responds to scale out events by launching instances and scale in events by terminating instances.

2. Auto Scaling puts the instance into a wait state (`Pending:Wait` or `Terminating:Wait`). The instance remains in this state until either you tell Auto Scaling to continue or the timeout period ends.

3. You can perform a custom action using one or more of the following options:

   - Define a CloudWatch Events target to invoke a Lambda function when a lifecycle action occurs. The Lambda function is invoked when Auto Scaling submits an event for a lifecycle action to CloudWatch Events. The event contains information about the instance that is launching or terminating, and a token that you can use to control the lifecycle action.

   - Define a notification target for the lifecycle hook. Auto Scaling sends a message to the notification target. The message contains information about the instance that is launching or terminating, and a token that you can use to control the lifecycle action.

   - Create a script that runs on the instance as the instance starts. The script can control the lifecycle action using the ID of the instance on which it runs.

4. By default, the instance remains in a wait state for one hour, and then Auto Scaling continues the launch or terminate process (`Pending:Proceed` or `Terminating:Proceed`). If you need more time, you can restart the timeout period by recording a heartbeat. If you finish before the timeout period ends, you can complete the lifecycle action, which continues the launch or termination process.

The following illustration shows the transitions between instance states in this process:
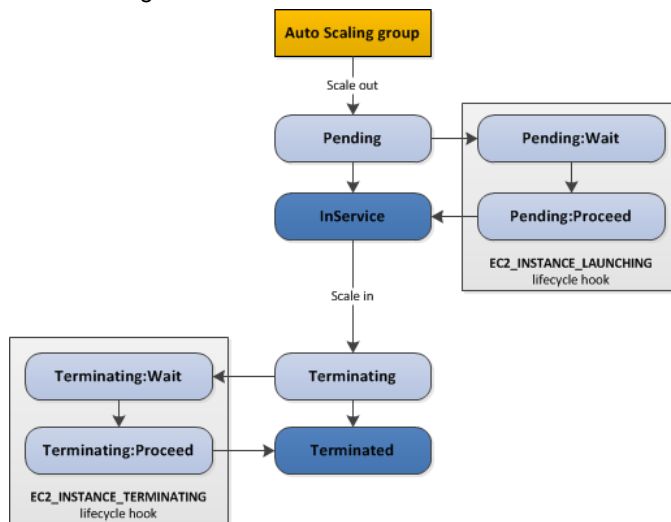
For more information about the complete lifecycle of instances in an Auto Scaling group, see Auto Scaling Lifecycle (p. 7).

Considerations When Using Lifecycle Hooks

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

Considerations

- Keeping Instances in a Wait State (p. 91)
- Cooldowns and Custom Actions (p. 91)
- Health Check Grace Period (p. 91)

Keeping Instances in a Wait State

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

- Set the heartbeat timeout for the lifecycle hook when you create the lifecycle hook. With the put-lifecycle-hook command, use the `--heartbeat-timeout` parameter. With the **PutLifecycleHook** operation, use the `HeartbeatTimeout` parameter.
- Continue to the next state if you finish before the timeout period ends, using the complete-lifecycle-action command or the **CompleteLifecycleAction** operation.
- Restart the timeout period by recording a heartbeat, using the record-lifecycle-action-heartbeat command or the **RecordLifecycleActionHeartbeat** operation. This increments the heartbeat timeout by the timeout value specified when you created the lifecycle hook. For example, if the timeout value is 1 hour, and you call this command after 30 minutes, the instance remains in a wait state for an additional hour, or a total of 90 minutes.

The maximum amount of time that you can keep an instance in a wait state is 48 hours or 100 times the heartbeat timeout, whichever is smaller.

Cooldowns and Custom Actions

When Auto Scaling launches or terminates an instance due to a simple scaling policy, a cooldown (p. 81) takes effect. The cooldown period helps ensure that the Auto Scaling group does not launch or terminate more instances than needed.

Consider an Auto Scaling group with a lifecycle hook that supports a custom action at instance launch. When the application experiences an increase in demand, Auto Scaling launches instances to add capacity. Because there is a lifecycle hook, the instance is put into the `Pending:Wait` state, which means that it is not available to handle traffic yet. When the instance enters the wait state, scaling actions due to simple scaling policies are suspended. When the instance enter the `InService` state, the cooldown period starts. When the cooldown period expires, any suspended scaling actions resume.

Health Check Grace Period

If you add a lifecycle hook to perform actions as your instances launch, the health check grace period does not start until you complete the lifecycle hook and the instance enters the `InService` state.

Lifecycle Action Result

At the conclusion of a lifecycle hook, the result is either `ABANDON` or `CONTINUE`.

If the instance is launching, `CONTINUE` indicates that your actions were successful, and that Auto Scaling can put the instance into service. Otherwise, `ABANDON` indicates that your custom actions were unsuccessful, and that Auto Scaling can terminate the instance.

If the instance is terminating, both `ABANDON` and `CONTINUE` allow the instance to terminate. However, `ABANDON` stops any remaining actions, such as other lifecycle hooks, while `CONTINUE` allows any other lifecycle hooks to complete.

Spot Instances

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

when a Spot Instance terminates, you must still complete the lifecycle action (using the **complete-lifecycle-action** command or the **CompleteLifecycleAction** operation).

Prepare for Notifications

You can optionally configure notifications when the instance enters a wait state, which enables you to perform a custom action. You can use Amazon CloudWatch Events, Amazon SNS, or Amazon SQS to receive the notifications. Choose whichever option you prefer.

Alternatively, if you have a script that configures your instances when they launch, you do not need to receive notification when the lifecycle action occurs. If you are not doing so already, update your script to retrieve the instance ID of the instance from the instance metadata. For more information, see Retrieving Instance Metadata.

Options
- Receive Notification Using CloudWatch Events (p. 92)
- Receive Notification Using Amazon SNS (p. 93)
- Receive Notification Using Amazon SQS (p. 93)

Receive Notification Using CloudWatch Events

You can use CloudWatch Events to set up a target to invoke a Lambda function when a lifecycle action occurs.

To set up notifications using CloudWatch Events

1. Create a Lambda function using the steps in Create a Lambda Function (p. 114) and note its Amazon Resource Name (ARN). For example, `arn:aws:lambda:us-west-2:123456789012:function:my-function`.
2. Create a CloudWatch Events rule that matches the lifecycle action using the following put-rule command:

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --
state ENABLED
```

   The `pattern.json` for an instance launch lifecycle action is:

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

   The `pattern.json` for an instance terminate lifecycle action is:

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ]
}
```

3. Create a target that invokes your Lambda function when the lifecycle action occurs, using the following put-targets command:

```
aws events put-targets --rule my-rule --targets
 Id=1,Arn=arn:aws:lambda:us-west-2:123456789012:function:my-function
```

4. When Auto Scaling responds to a scale out or scale in event, it puts the instance in a wait state. While the instance is in a wait state, the Lambda function is invoked. For more information about the event data, see Auto Scaling Events (p. 111).

Receive Notification Using Amazon SNS

You can use Amazon SNS to set up a notification target to receive notifications when a lifecycle action occurs.

To set up notifications using Amazon SNS

1. Create the target using Amazon SNS. For more information, see Create a Topic in the *Amazon Simple Notification Service Developer Guide*. Note the ARN of the target (for example, `arn:aws:sns:us-west-2:123456789012:my-sns-topic`).

2. Create an IAM role to grant Auto Scaling permission to access your notification target, using the steps in Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*. When prompted to select a role type, select **AWS Service Roles**, **AutoScaling Notification Access**. Note the ARN of the role. For example, `arn:aws:iam::123456789012:role/my-notification-role`.

3. When Auto Scaling responds to a scale out or scale in event, it puts the instance in a wait state. While the instance is in a wait state, Auto Scaling publishes a message to the notification target. The message includes the following event data:

   - **LifecycleActionToken** — The lifecycle action token.
   - **AccountId** — The AWS account ID.
   - **AutoScalingGroupName** — The name of the Auto Scaling group.
   - **LifecycleHookName** — The name of the lifecycle hook.
   - **EC2InstanceId** — The ID of the EC2 instance.
   - **LifecycleTransition** — The lifecycle hook type.

   For example:

   ```
   Service: AWS Auto Scaling
   Time: 2016-09-30T20:42:11.305Z
   RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
   LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
   AccountId: 123456789012
   AutoScalingGroupName: my-asg
   LifecycleHookName: my-hook
   EC2InstanceId: i-0598c7d356eba48d7
   LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
   NotificationMetadata: null
   ```

Receive Notification Using Amazon SQS

You can use Amazon SQS to set up a notification target to receive notifications when a lifecycle action occurs.

To set up notifications using Amazon SQS

1. Create the target using Amazon SQS. For more information, see Getting Started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*. Note the ARN of the target.

2. Create an IAM role to grant Auto Scaling permission to access your notification target, using the steps in Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*. When prompted to select a role type, select **AWS Service Roles**, **AutoScaling Notification Access**. Note the ARN of the role. For example, `arn:aws:iam::123456789012:role/my-notification-role`.

3. When Auto Scaling responds to a scale out or scale in event, it puts the instance in a wait state. While the instance is in a wait state, Auto Scaling publishes a message to the notification target.

Add Lifecycle Hooks

Create one or more lifecycle hooks using the put-lifecycle-hook command:

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-hook --auto-
scaling-group-name my-asg
```

To perform an action on scale out, add the following option:

```
--lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

To perform an action on scale in, add the following option instead:

```
--lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

(Optional) If you are using a notification, add the following options:

```
--notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
--role-arn arn:aws:iam::123456789012:role/my-notification-role
```

Each Auto Scaling group can have multiple lifecycle hooks. However, there is a limit on the number of hooks per Auto Scaling group. For more information, see Auto Scaling Account Limits.

Complete the Lifecycle Hook

When Auto Scaling responds to a scale out or scale in event, it puts the instance in a wait state and sends any notifications. Auto Scaling continues the launch or terminate process after you complete the lifecycle hook.

To complete a lifecycle hook

1. While the instance is in a wait state, you can perform a custom action. For more information, see Prepare for Notifications (p. 92).

2. If you need more time to complete the custom action, use the record-lifecycle-action-heartbeat command to restart the timeout period and keep the instance in a wait state. You can specify the lifecycle action token you received in the previous step, as shown in the following command:

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

Alternatively, you can specify the ID of the instance you retrieved in the previous step, as shown in the following command:

```
aws autoscaling record-lifecycle-action-heartbeat --instance-id i-1a2b3c4d
 --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg
```

3. If you finish the custom action before the timeout period ends, use the complete-lifecycle-action command so that Auto Scaling can continue launching or terminating the instance. Note that you can specify the lifecycle action token, as shown in the following command:

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

Alternatively, you can specify the ID of the instance, as shown in the following command:

```
aws autoscaling complete-lifecycle-action --lifecycle-action-result
 CONTINUE --instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook --
auto-scaling-group-name my-asg
```

Test the Notification

To generate a notification for a launch event, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. Auto Scaling launches the EC2 instance, and you'll receive a notification within a few minutes.

To change the desired capacity using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Details** tab, choose **Edit**.
5. For **Desired**, increase the current value by 1. Note that if this value exceeds **Max**, you must also increase the value of **Max** by 1.
6. Choose **Save**.
7. After a few minutes, you'll receive notification for the event. If you do not need the additional instance that you launched for this test, you can decrease **Desired** by 1. After a few minutes, you'll receive notification for the event.

Temporarily Removing Instances from Your Auto Scaling Group

Auto Scaling enables you to put an instance that is in the `InService` state into the `Standby` state, update or troubleshoot the instance, and then return the instance to service. Instances that are on standby are still part of the Auto Scaling group, but they do not actively handle application traffic.

Important
You are billed for instances that are in a standby state.

For example, you can change the launch configuration for an Auto Scaling group at any time, and any subsequent instances that the Auto Scaling group launches use this configuration. However, the Auto Scaling group does not update the instances that are currently in service. You can either terminate

these instances and let the Auto Scaling group replace them, or you can put the instances on standby, update the software, and then put the instances back in service.

Contents

How the Standby State Works

The standby state works as follows to help you temporarily remove an instance from your Auto Scaling group:

1. You put the instance into the standby state. The instance remains in this state until you exit the standby state.
2. If there is a load balancer or target group attached to your Auto Scaling group, the instance is deregistered from the load balancer or target group.
3. By default, Auto Scaling decrements the desired capacity of your Auto Scaling group when you put an instance on standby. This prevents Auto Scaling from launching an additional instance while you have this instance on standby. Alternatively, you can specify that Auto Scaling does not decrement the capacity. This causes Auto Scaling to launch an additional instance to replace the one on standby.
4. You can update or troubleshoot the instance.
5. You return the instance to service by exiting the standby state.
6. Auto Scaling increments the desired capacity when you put an instance that was on standby back in service. If you did not decrement the capacity when you put the instance on standby, Auto Scaling detects that you have more instances than you need, and applies the termination policy in effect to reduce the size of your Auto Scaling group. For more information, see Controlling Which Instances Auto Scaling Terminates During Scale In (p. 84).
7. If there is a load balancer or target group attached to your Auto Scaling group, the instance is registered with the load balancer or target group.

The following illustration shows the transitions between instance states in this process:
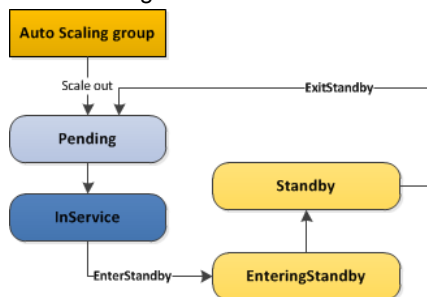
For more information about the complete lifecycle of instances in an Auto Scaling group, see Auto Scaling Lifecycle (p. 7).

Health Status of an Instance in a Standby State

Auto Scaling does not perform health checks on instances that are in a standby state. While the instance is in a standby state, its health status reflects the status that it had before you put it on standby. Auto Scaling does not perform a health check on the instance until you put it back in service.

For example, if you put a healthy instance on standby and then terminate it, Auto Scaling continues to report the instance as healthy. If you return the terminated instance to service, Auto Scaling performs a health check on the instance, determines that it is unhealthy, and launches a replacement instance.

Temporarily Remove an Instance Using the AWS Management Console

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group.
4. On the **Instances** tab, select the instance.
5. Choose **Actions**, **Set to Standby**.
6. On the **Set to Standby** page, select the checkbox if you want Auto Scaling to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Choose **Set to Standby**.
7. You can update or troubleshoot your instance as needed. When you have finished, continue with the next step to return the instance to service.
8. Select the instance, choose **Actions**, **Set to InService**. On the **Set to InService** page, choose **Set to InService**.

Temporarily Remove an Instance Using the AWS CLI

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance using the AWS CLI

1. Use the following describe-auto-scaling-instances command to identify the instance to update:

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

The following is an example response:

```
{
    "AutoScalingInstances": [
        {
            "AvailabilityZone": "us-west-2a",
            "InstanceId": "i-5b73d709",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService",
            "LaunchConfigurationName": "my-lc"
        },
        ...
    ]
```

```
}
```

2. Move the instance into a `Standby` state using the following enter-standby command. The `--should-decrement-desired-capacity` option decreases the desired capacity so that Auto Scaling does not launch a replacement instance.

```
aws autoscaling enter-standby --instance-ids i-5b73d709 --auto-scaling-
group-name my-asg --should-decrement-desired-capacity
```

The following is an example response:

```
{
    "Activities": [
        {
            "Description": "Moving EC2 instance to Standby: i-5b73d709",
            "AutoScalingGroupName": "my-asg",
            "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
            "Details": "{\"Availability Zone\":\"us-west-2a\"}",
            "StartTime": "2014-12-15T21:31:26.150Z",
            "Progress": 50,
            "Cause": "At 2014-12-15T21:31:26Z instance i-5b73d709 was
 moved to standby
                in response to a user request, shrinking the capacity from 4
 to 3.",
            "StatusCode": "InProgress"
        }
    ]
}
```

3. (Optional) Verify that the instance is in `Standby` using the following describe-auto-scaling-instances command:

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-5b73d709
```

The following is an example response. Notice that the status of the instance is now `Standby`.

```
{
    "AutoScalingInstances": [
        {
            "AvailabilityZone": "us-west-2a",
            "InstanceId": "i-5b73d709",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby",
            "LaunchConfigurationName": "my-lc"
        }
    ]
}
```

4. You can update or troubleshoot your instance as needed. When you have finished, continue with the next step to return the instance to service.

5. Put the instance back in service using the following exit-standby command:

```
aws autoscaling exit-standby --instance-ids i-5b73d709 --auto-scaling-
group-name my-asg
```

The following is an example response:

```
{
    "Activities": [
        {
            "Description": "Moving EC2 instance out of Standby:
 i-5b73d709",
            "AutoScalingGroupName": "my-asg",
            "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
            "Details": "{\"Availability Zone\":\"us-west-2a\"}",
            "StartTime": "2014-12-15T21:46:14.678Z",
            "Progress": 30,
            "Cause": "At 2014-12-15T21:46:14Z instance i-5b73d709 was
 moved out of standby in
                response to a user request, increasing the capacity from 3
 to 4.",
            "StatusCode": "PreInService"
        }
    ]
}
```

6. (Optional) Verify that the instance is back in service using the following `describe-auto-scaling-instances` command:

```
aws autoscaling describe-auto-scaling-instances --instance-ids i-5b73d709
```

The following is an example response. Notice that the status of the instance is `InService`.

```
{
    "AutoScalingInstances": [
        {
            "AvailabilityZone": "us-west-2a",
            "InstanceId": "i-5b73d709",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService",
            "LaunchConfigurationName": "my-lc"
        }
    ]
}
```

Suspending and Resuming Auto Scaling Processes

Auto Scaling enables you to suspend and then resume one or more of the Auto Scaling processes in your Auto Scaling group. This can be very useful when you want to investigate a configuration problem or other issue with your web application and then make changes to your application, without triggering the Auto Scaling process.

Auto Scaling might suspend processes for Auto Scaling groups that repeatedly fail to launch instances. This is known as an *administrative suspension*, and most commonly applies to Auto Scaling groups that have been trying to launch instances for over 24 hours but have not succeeded in launching any instances. You can resume processes suspended for administrative reasons.

Contents

Auto Scaling Processes

Auto Scaling supports the following processes:

Launch

Adds a new EC2 instance to the group, increasing its capacity.

Warning

If you suspend `Launch`, this disrupts other processes. For example, you can't return an instance in a standby state to service if the `Launch` process is suspended, because the group can't scale.

Terminate

Removes an EC2 instance from the group, decreasing its capacity.

Warning

If you suspend `Terminate`, this disrupts other processes.

HealthCheck

Checks the health of the instances. Auto Scaling marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing tells Auto Scaling that the instance is unhealthy. This process can override the health status of an instance that you set manually.

ReplaceUnhealthy

Terminates instances that are marked as unhealthy and subsequently creates new instances to replace them. This process works with the `HealthCheck` process, and uses both the `Terminate` and `Launch` processes.

AZRebalance

Balances the number of EC2 instances in the group across the Availability Zones in the region. If you remove an Availability Zone from your Auto Scaling group or an Availability Zone otherwise becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone before terminating the unhealthy or unavailable instances. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the instances evenly across the Availability Zones for the group.

Note that if you suspend `AZRebalance` and a scale out or scale in event occurs, Auto Scaling still tries to balance the Availability Zones. For example, during scale out, Auto Scaling launches the instance in the Availability Zone with the fewest instances.

If you suspend `Launch`, `AZRebalance` neither launches new instances nor terminates existing instances. This is because `AZRebalance` terminates instances only after launching the replacement instances. If you suspend `Terminate`, your Auto Scaling group can grow up to ten percent larger than its maximum size, because Auto Scaling allows this temporarily during rebalancing activities. If Auto Scaling cannot terminate instances, your Auto Scaling group could remain above its maximum size until you resume the `Terminate` process.

AlarmNotification

Accepts notifications from CloudWatch alarms that are associated with the group.

If you suspend `AlarmNotification`, Auto Scaling does not automatically execute policies that would be triggered by an alarm. If you suspend `Launch` or `Terminate`, Auto Scaling would not be able to execute scale out or scale in policies, respectively.

ScheduledActions

Performs scheduled actions that you create.

If you suspend `Launch` or `Terminate`, scheduled actions that involve launching or terminating instances are affected.

`AddToLoadBalancer`

Adds instances to the attached load balancer or target group when they are launched.

If you suspend `AddToLoadBalancer`, Auto Scaling launches the instances but does not add them to the load balancer or target group. If you resume the `AddToLoadBalancer` process, Auto Scaling resumes adding instances to the load balancer or target group when they are launched. However, Auto Scaling does not add the instances that were launched while this process was suspended. You must register those instances manually.

Suspend and Resume Processes Using the Console

You can suspend and resume individual processes using the AWS Management Console.

To suspend and resume processes using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group.
4. On the **Details** tab, choose **Edit**.
5. For **Suspended Processes**, select the process to suspend.
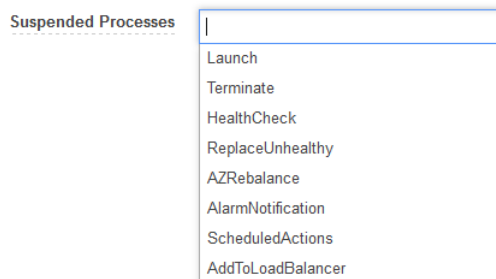
To resume a suspended process, remove it from **Suspended Processes**.
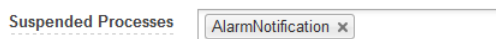
6. Choose **Save**.

Suspend and Resume Processes Using the AWS CLI

You can suspend and resume individual processes or all processes.

To suspend a process

Use the suspend-processes command with the `--scaling-processes` option as follows:

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-processes AlarmNotification
```

To suspend all processes

Use the suspend-processes command as follows (omitting the `--scaling-processes` option):

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

To resume a suspended process

Use the resume-processes command as follows:

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-
processes AlarmNotification
```

To resume all suspended processes

Use the resume-processes command as follows (omitting the --scaling-processes option):

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

Monitoring Your Auto Scaling Instances and Groups

You can use the following features to monitor your Auto Scaling instances and groups.

Health checks

Auto Scaling periodically performs health checks on the instances in your Auto Scaling group and identifies any instances that are unhealthy. You can configure Auto Scaling to determine the health status of an instance using Amazon EC2 status checks, Elastic Load Balancing health checks, or custom health checks. For more information, see Health Checks for Auto Scaling Instances (p. 104).

CloudWatch metrics

Auto Scaling publishes data points to Amazon CloudWatch about your Auto Scaling groups. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. You can use these metrics to verify that your system is performing as expected. For more information, see Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch (p. 105).

CloudWatch Events

Auto Scaling can submit events to Amazon CloudWatch Events when your Auto Scaling groups launch or terminate instances, or when a lifecycle action occurs. This enables you to invoke a Lambda function when the event occurs. For more information, see Getting CloudWatch Events When Your Auto Scaling Group Scales (p. 110).

SNS notifications

Auto Scaling can send Amazon SNS notifications when your Auto Scaling groups launch or terminate instances. For more information, see Getting SNS Notifications When Your Auto Scaling Group Scales (p. 116).

CloudTrail logs

AWS CloudTrail enables you to keep track of the calls made to the Auto Scaling API by or on behalf of your AWS account. CloudTrail stores the information in log files in the Amazon S3 bucket that you specify. You can use these log files to monitor activity of your Auto Scaling groups by determining which requests were made, the source IP addresses where the requests came from, who made the request, when the request was made, and so on. For more information, see Logging Auto Scaling API Calls By Using AWS CloudTrail (p. 121).

Health Checks for Auto Scaling Instances

Auto Scaling periodically performs health checks on the instances in your Auto Scaling group and identifies any instances that are unhealthy. After Auto Scaling marks an instance as unhealthy, it is scheduled for replacement. For more information, see Replacing Unhealthy Instances (p. 58).

Instance Health Status

An Auto Scaling instance is either healthy or unhealthy. Auto Scaling determines the health status of an instance using one or more of the following:

- Status checks provided by Amazon EC2. For more information, see Status Checks for Your Instances in the *Amazon EC2 User Guide for Linux Instances*.
- Health checks provided by Elastic Load Balancing. For more information, see Health Checks for Your Target Groups in the *Application Load Balancer Guide* or Configure Health Checks for Your Classic Load Balancer in the *Classic Load Balancer Guide*.
- Custom health checks. For more information, see Instance Health Status and Custom Health Checks (p. 104).

By default, Auto Scaling health checks use the results of the status checks to determine the health status of an instance. Auto Scaling marks an instance as unhealthy if its instance status is any value other than `running` or its system status is `impaired`.

If you have attached a load balancer to your Auto Scaling group, you can optionally have Auto Scaling include the results of Elastic Load Balancing health checks when determining the health status of an instance. After you add these health checks, Auto Scaling also marks an instance as unhealthy if Elastic Load Balancing reports the instance state as `OutOfService`. For more information, see Adding Health Checks to Your Auto Scaling Group (p. 49).

Health Check Grace Period

Frequently, an Auto Scaling instance that has just come into service needs to warm up before it can pass the Auto Scaling health check. Auto Scaling waits until the health check grace period ends before checking the health status of the instance. While the EC2 status checks and ELB health checks can complete before the health check grace period expires, Auto Scaling does not act on them until the health check grace period expires. To provide ample warm-up time for your instances, ensure that the health check grace period covers the expected startup time for your application. Note that if you add a lifecycle hook to perform actions as your instances launch, the health check grace period does not start until the lifecycle hook is completed and the instance enters the `InService` state.

Instance Health Status and Custom Health Checks

If you have custom health checks, you can send the information from your health checks to Auto Scaling so that Auto Scaling can use this information. For example, if you determine that an instance is not functioning as expected, you can set the health status of the instance to `Unhealthy`. The next time that Auto Scaling performs a health check on the instance, it will determine that the instance is unhealthy and then launch a replacement instance.

Use the following set-instance-health command to set the health state of the specified instance to `Unhealthy`:

```
aws autoscaling set-instance-health --instance-id i-123abc45d --health-status
 Unhealthy
```

Use the following `describe-auto-scaling-groups` command to verify that the instance state is `Unhealthy`:

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

The following is an example response that shows that the health status of the instance is `Unhealthy` and that the instance is terminating:

```
{
    "AutoScalingGroups": [
        {
            ....
            "Instances": [
                {
                    "InstanceId": "i-123abc45d",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Unhealthy",
                    "LifecycleState": "Terminating",
                    "LaunchConfigurationName": "my-lc"
                },
                ...
            ]
        }
    ]
}
```

Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch

Amazon CloudWatch enables you to retrieve statistics as an ordered set of time-series data, known as metrics. You can use these metrics to verify that your system is performing as expected.

Amazon EC2 sends metrics to CloudWatch that describe your Auto Scaling instances. These metrics are available for any EC2 instance, not just those in an Auto Scaling group. For more information, see Instance Metrics in the *Amazon EC2 User Guide for Linux Instances*.

Auto Scaling groups can send metrics to CloudWatch that describe the group itself. You must enable these metrics.

Contents

Auto Scaling Group Metrics

The `AWS/AutoScaling` namespace includes the following metrics.

Metric	Description
`GroupMinSize`	The minimum size of the Auto Scaling group.
`GroupMaxSize`	The maximum size of the Auto Scaling group.
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain.
`GroupInServiceInstances`	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating.
`GroupPendingInstances`	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating.
`GroupStandbyInstances`	The number of instances that are in a `Standby` state. Instances in this state are still running but are not actively in service.
`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

To filter the metrics for your Auto Scaling group by group name, use the `AutoScalingGroupName` dimension.

Enable Auto Scaling Group Metrics

When you enable Auto Scaling group metrics, Auto Scaling sends aggregated data to CloudWatch every minute.

To enable group metrics using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. In the navigation pane, choose **Auto Scaling Groups**.

3. Select your Auto Scaling group.

4. On the **Monitoring** tab, for **Auto Scaling Metrics**, choose **Enable Group Metrics Collection**. If you don't see this option, select **Auto Scaling** for **Display**.

To disable group metrics using the console
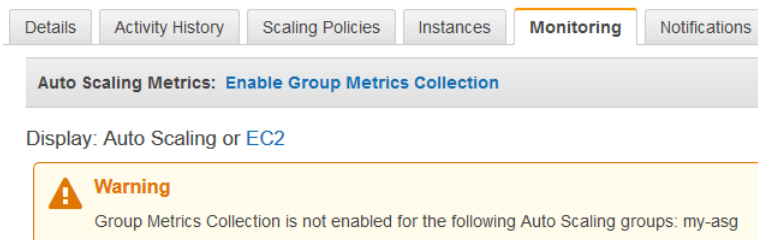
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Monitoring** tab, for **Auto Scaling Metrics**, choose **Disable Group Metrics Collection**. If you don't see this option, select **Auto Scaling** for **Display**.
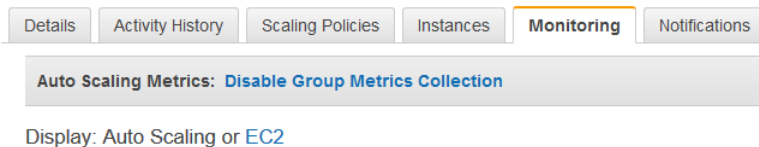
To enable group metrics using the AWS CLI

Enable one or more group metrics using the enable-metrics-collection command. For example, the following command enables the GroupDesiredCapacity metric.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg --
metrics GroupDesiredCapacity --granularity "1Minute"
```

If you omit the --metrics option, all metrics are enabled.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg --
granularity "1Minute"
```

To disable group metrics using the AWS CLI

Use the disable-metrics-collection command. For example, the following command disables all Auto Scaling group metrics:

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

Enable Auto Scaling Instance Metrics

You can enable basic or detailed monitoring for the instances in your Auto Scaling group when you create a launch configuration. By default, basic monitoring is enabled when you create the launch configuration using the AWS Management Console and detailed monitoring is enabled when you create the launch configuration using the AWS CLI or an API.

If you have an Auto Scaling group and need to change which type of monitoring is enabled for your Auto Scaling instances, you must create a new launch configuration and update the Auto Scaling group to use this launch configuration.

To enable Auto Scaling instance metrics using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Create a launch configuration as follows:

   a. In the navigation pane, choose **Launch Configurations**.
   b. Choose **Create launch configuration**.
   c. On the first two pages, choose and AMI and an instance type.

    d.   On the **Configure details** page, to enable detailed monitoring, select **Enable CloudWatch detailed monitoring**. Otherwise, basic monitoring is enabled.

    e.   Complete the wizard to create your launch configuration.

3.   Create an Auto Scaling group using this launch configuration as follows:

    a.   In the navigation pane, choose **Auto Scaling Groups**.

    b.   Choose **Create Auto Scaling group**.

    c.   On the first page, select **Create an Auto Scaling group from an existing launch configuration**, select the launch configuration, and then choose **Next Step**.

    d.   Complete the wizard to create your Auto Scaling group.

4.   If you have an existing Auto Scaling group, you can update it to use this launch configuration as follows:

    a.   In the navigation pane, choose **Auto Scaling Groups**.

    b.   Select the Auto Scaling group.

    c.   On the **Details** tab, choose **Edit**.

    d.   For **Launch Configuration**, select the launch configuration.

    e.   Choose **Save**.

    f.   From now on, the instances that the Auto Scaling group launches will use the updated monitoring type. However, if you have existing instances in the Auto Scaling group, they maintain the previous monitoring type. You can terminate these instances so that Auto Scaling replaces them, or update each instance individually.

    g.   If you have CloudWatch alarms associated with your Auto Scaling group, update each alarm each alarm so that its period matches the monitoring type (300 seconds for basic monitoring and 60 seconds for detailed monitoring).

       If you change from detailed monitoring to basic monitoring but do not update your alarms to match the five-minute data aggregations, they continue to check for statistics every minute and might find no data available for as many as four out of every five periods.

To enable monitoring using the AWS CLI

1.   Use the create-launch-configuration command with the `--instance-monitoring` option. Set this option to `true` to enable detailed monitoring or `false` to enable basic monitoring.

```
--instance-monitoring Enabled=true
```

2.   Use the update-auto-scaling-group command with the `--launch-configuration-name` option to use this launch configuration.

```
--launch-configuration-name my-lc
```

3.   Use the put-metric-alarm command with the `--period` option to ensure the period matches the monitoring type (300 seconds for basic monitoring and 60 seconds for detailed monitoring).

```
--period 300
```

View CloudWatch Metrics

You can view the CloudWatch metrics for your Auto Scaling groups and instances using the Amazon EC2 console. These metrics are displayed as monitoring graphs.

Alternatively, you can view these metrics using the CloudWatch console.

To view metrics using the Amazon EC2 console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. Choose the **Monitoring** tab.
5. (Optional) To filter the results by time, select a time range from **Showing data for**.
6. To view the metrics for your groups, for **Display**, choose **Auto Scaling**. To get a larger view of a single metric, select its graph. The following metrics are available for groups:

   - Minimum Group Size — `GroupMinSize`
   - Maximum Group Size — `GroupMaxSize`
   - Desired Capacity — `GroupDesiredCapacity`
   - In Service Instances — `GroupInServiceInstances`
   - Pending Instances — `GroupPendingInstances`
   - Standby Instances — `GroupStandbyInstances`
   - Terminating Instances — `GroupTerminatingInstances`
   - Total Instances — `GroupTotalInstances`

7. To view metrics for your instances, for **Display**, choose **EC2**. To get a larger view of a single metric, select its graph. The following metrics are available for instances:

   - CPU Utilization — `CPUUtilization`
   - Disk Reads — `DiskReadBytes`
   - Disk Read Operations — `DiskReadOps`
   - Disk Writes — `DiskWriteBytes`
   - Disk Write Operations — `DiskWriteOps`
   - Network In — `NetworkIn`
   - Network Out — `NetworkOut`
   - Status Check Failed (Any) — `StatusCheckFailed`
   - Status Check Failed (Instance) — `StatusCheckFailed_Instance`
   - Status Check Failed (System) — `StatusCheckFailed_System`

To view metrics using the CloudWatch console

For more information, see Aggregate Statistics by Auto Scaling Group.

To view CloudWatch metrics using the AWS CLI

To view all metrics for all your Auto Scaling groups, use the following list-metrics command:

```
aws cloudwatch list-metrics --namespace "AWS/AutoScaling"
```

To view the metrics for a single Auto Scaling group, specify the `AutoScalingGroupName` dimension as follows:

```
aws cloudwatch list-metrics --namespace "AWS/AutoScaling" --dimensions
 Name=AutoScalingGroupName,Value=my-asg
```

To view a single metric for all your Auto Scaling groups, specify the name of the metric as follows:

```
aws cloudwatch list-metrics --namespace "AWS/AutoScaling" --metric-name
 GroupDesiredCapacity
```

Create Amazon CloudWatch Alarms

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

An alarm has three possible states:

- `OK`— The value of the metric remains within the range that you've specified.
- `ALARM`— The value of the metric is out of the range that you've specified for a specified time duration.
- `INSUFFICIENT_DATA`— The metric is not yet available or there is not enough data available to determine the alarm state.

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

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

To create a CloudWatch alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. On the navigation pane, choose **Alarms**.
3. Choose **Create Alarm**.
4. Choose the **EC2 Metrics** category.
5. (Optional) You can filter the results. To see the instance metrics, choose **Per-Instance Metrics**. To see the Auto Scaling group metrics, choose **By Auto Scaling Group**.
6. Select a metric, and then choose **Next**.
7. Specify a threshold for the alarm and the action to take.

   For more information, see Creating CloudWatch Alarms in the *Amazon CloudWatch User Guide*.
8. Choose **Create Alarm**.

Getting CloudWatch Events When Your Auto Scaling Group Scales

When you use Auto Scaling to scale your applications automatically, it is useful to know when Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. You can configure Auto Scaling to send events to Amazon CloudWatch Events whenever your Auto Scaling group scales.

For more information, see the Amazon CloudWatch Events User Guide.

Contents

Auto Scaling Events

Auto Scaling supports sending events to CloudWatch Events when the following events occur:

EC2 Instance-launch Lifecycle Action

Auto Scaling moved an instance to a `Pending:Wait` state due to a lifecycle hook.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-12345678",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_LAUNCHING"
  }
}
```

EC2 Instance Launch Successful

Auto Scaling successfully launched an instance.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
```

```
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "us-west-2",
    "resources": [
        "auto-scaling-group-arn",
        "intance-arn"
    ],
    "detail": {
        "StatusCode": "InProgress",
        "Description": "Launching a new EC2 instance: i-12345678",
        "AutoScalingGroupName": "my-auto-scaling-group",
        "ActivityId": "87654321-4321-4321-4321-210987654321",
        "Details": {
            "Availability Zone": "us-west-2b",
            "Subnet ID": "subnet-12345678"
        },
        "RequestId": "12345678-1234-1234-1234-123456789012",
        "StatusMessage": "",
        "EndTime": "yyyy-mm-ddThh:mm:ssZ",
        "EC2InstanceId": "i-12345678",
        "StartTime": "yyyy-mm-ddThh:mm:ssZ",
        "Cause": "description-text",
    }
}
```

EC2 Instance Launch Unsuccessful

Auto Scaling failed to launch an instance.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "intance-arn"
  ],
  "detail": {
      "StatusCode": "Failed",
      "AutoScalingGroupName": "my-auto-scaling-group",
      "ActivityId": "87654321-4321-4321-4321-210987654321",
      "Details": {
          "Availability Zone": "us-west-2b",
          "Subnet ID": "subnet-12345678"
      },
      "RequestId": "12345678-1234-1234-1234-123456789012",
      "StatusMessage": "message-text",
      "EndTime": "yyyy-mm-ddThh:mm:ssZ",
      "EC2InstanceId": "i-12345678",
      "StartTime": "yyyy-mm-ddThh:mm:ssZ",
```

```
      "Cause": "description-text",
  }
}
```

EC2 Instance-terminate Lifecycle Action

Auto Scaling moved an instance to a `Terminating:Wait` state due to a lifecycle hook.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-12345678",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING"
  }
}
```

EC2 Instance Terminate Successful

Auto Scaling successfully terminated an instance.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "intance-arn"
  ],
  "detail": {
      "StatusCode": "InProgress",
      "Description": "Terminating EC2 instance: i-12345678",
      "AutoScalingGroupName": "my-auto-scaling-group",
      "ActivityId": "87654321-4321-4321-4321-210987654321",
```

```
        "Details": {
            "Availability Zone": "us-west-2b",
            "Subnet ID": "subnet-12345678"
        },
        "RequestId": "12345678-1234-1234-1234-123456789012",
        "StatusMessage": "",
        "EndTime": "yyyy-mm-ddThh:mm:ssZ",
        "EC2InstanceId": "i-12345678",
        "StartTime": "yyyy-mm-ddThh:mm:ssZ",
        "Cause": "description-text",
    }
}
```

EC2 Instance Terminate Unsuccessful

Auto Scaling failed to terminate an instance.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "intance-arn"
  ],
  "detail": {
      "StatusCode": "Failed",
      "AutoScalingGroupName": "my-auto-scaling-group",
      "ActivityId": "87654321-4321-4321-4321-210987654321",
      "Details": {
          "Availability Zone": "us-west-2b",
          "Subnet ID": "subnet-12345678"
      },
      "RequestId": "12345678-1234-1234-1234-123456789012",
      "StatusMessage": "message-text",
      "EndTime": "yyyy-mm-ddThh:mm:ssZ",
      "EC2InstanceId": "i-12345678",
      "StartTime": "yyyy-mm-ddThh:mm:ssZ",
      "Cause": "description-text",
  }
}
```

Create a Lambda Function

Use the following procedure to create a Lambda function to handle an Auto Scaling event.

To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.

2. If you are new to Lambda, you see a welcome page; choose **Get Started Now**; otherwise, choose **Create a Lambda function**.

3. On the **Select blueprint** page, type `hello-world` for **Filter**, and then select the **hello-world** blueprint.

4. On the **Configure triggers** page, choose **Next**.

5. On the **Configure function** page, do the following:

    a. Type a name and description for the Lambda function.

    b. Edit the code for the Lambda function. For example, the following code simply logs the event:

    ```
    console.log('Loading function');

    exports.handler = function(event, context) {
        console.log("AutoScalingEvent()");
        console.log("Event data:\n" + JSON.stringify(event, null, 4));
        context.succeed("...");
    };
    ```

    c. For **Role**, choose **Choose an existing role** if you have an existing role that you'd like to use, and then choose your role from **Existing role**. Alternatively, to create a new role, choose one of the other options for **Role** and then follow the directions.

    d. (Optional) For **Advanced settings**, make any changes that you need.

    e. Choose **Next**.

6. On the **Review** page, choose **Create function**.

Route Events to Your Lambda Function

Use the following procedure to route Auto Scaling events to your Lambda function.

To route events to your Lambda function

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. On the navigation pane, choose **Events**.

3. Choose **Create rule**.

4. For **Event selector**, choose **Auto Scaling** as the event source. By default, the rule applies to all Auto Scaling events for all of your Auto Scaling groups. Alternatively, you can select specific events or a specific Auto Scaling group.

5. For **Targets**, choose **Add target**. Choose **Lambda function** as the target type, and then select your Lambda function.

6. Choose **Configure details**.

7. For **Rule definition**, type a name and description for your rule and then choose **Create rule**.

To test your rule, change the size of your Auto Scaling group. If you used the example code for your Lambda function, it logs the event to CloudWatch Logs.

To test your rule

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. On the navigation pane, choose **Auto Scaling Groups**, and then select your Auto Scaling group.

3. On the **Details** tab, choose **Edit**.

4. Change the value of **Desired**, and then choose **Save**.

5. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

6. On the navigation pane, choose **Logs**.
7. Select the log group for your Lambda function (for example, **/aws/lambda/**_my-function_).
8. Select a log stream to view the event data. The data is displayed, similar to the following:

```
Event Data
▼ 2016-02-22T17:48:20.778Z ea1fjqinxq6pwo9d Loading function
▼ START RequestId: 7560439b-d98c-11e5-932d-f52757e7aee0 Version: $LATEST
▼ 2016-02-22T17:48:20.813Z 7560439b-d98c-11e5-932d-f52757e7aee0 AutoScalingEvent()
▼ 2016-02-22T17:48:20.814Z 7560439b-d98c-11e5-932d-f52757e7aee0 Event data:
{
    "version": "0",
    "id": "df9b0c8c-89c8-4748-92cb-ac68a9029ada",
    "detail-type": "EC2 Instance Launch Successful",
    "source": "aws.autoscaling",
```

Getting SNS Notifications When Your Auto Scaling Group Scales

When you use Auto Scaling to scale your applications automatically, it is useful to know when Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. Amazon SNS coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. You can configure Auto Scaling to send an SNS notification whenever your Auto Scaling group scales.

Amazon SNS can deliver notifications as HTTP or HTTPS POST, email (SMTP, either plain-text or in JSON format), or as a message posted to an Amazon SQS queue. For more information, see What Is Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

For example, if you configure your Auto Scaling group to use the `autoscaling:EC2_INSTANCE_TERMINATE` notification type, and your Auto Scaling group terminates an instance, it sends an email notification. This email contains the details of the terminated instance, such as the instance ID and the reason that the instance was terminated.

> **Tip**
> If you prefer, you can use Amazon CloudWatch Events to configure a target to invoke a Lambda function when your Auto Scaling group scales or when a lifecycle action occurs. For more information, see Getting CloudWatch Events When Your Auto Scaling Group Scales (p. 110).

Contents

SNS Notifications

Auto Scaling supports sending Amazon SNS notifications when the following events occur.

Event	Description
`autoscaling:EC2_INSTANCE_LAUNCH`	Successful instance launch
`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`	Failed instance launch
`autoscaling:EC2_INSTANCE_TERMINATE`	Successful instance termination

Event	Description
`autoscaling:EC2_INSTANCE_TERMINATE_ERROR`	Failed instance termination

The message includes the following information:

- **Event** — The event.
- **AccountId** — The AWS account ID.
- **AutoScalingGroupName** — The name of the Auto Scaling group.
- **AutoScalingGroupARN** — The ARN of the Auto Scaling group.
- **EC2InstanceId** — The ID of the EC2 instance.

For example:

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup
 constraints to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-c9663da0","Availability Zone":"us-west-2b"}
```

Configure Amazon SNS

To use Amazon SNS to send email notifications, you must first create a *topic* and then subscribe your email addresses to the topic.

Create an Amazon SNS Topic

An SNS topic is a logical access point, a communication channel your Auto Scaling group uses to send the notifications. You create a topic by specifying a name for your topic.

For more information, see Create a Topic in the *Amazon Simple Notification Service Developer Guide*.

Subscribe to the Amazon SNS Topic

To receive the notifications that your Auto Scaling group sends to the topic, you must subscribe an endpoint to the topic. In this procedure, for **Endpoint**, specify the email address where you want to receive the notifications from Auto Scaling.

For more information, see Subscribe to a Topic in the *Amazon Simple Notification Service Developer Guide*.

Confirm Your Amazon SNS Subscription

Amazon SNS sends a confirmation email to the email address you specified in the previous step.

Make sure you open the email from AWS Notifications and choose the link to confirm the subscription before you continue with the next step.
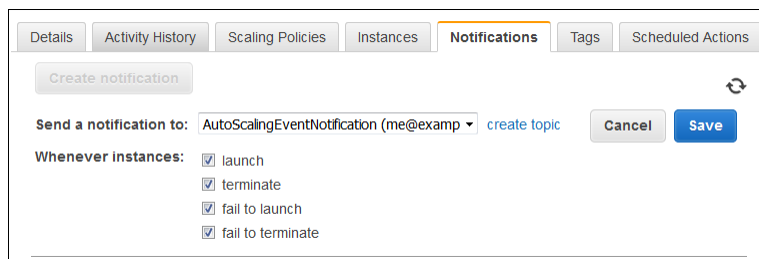
You will receive an acknowledgement message from AWS. Amazon SNS is now configured to receive notifications and send the notification as an email to the email address that you specified.

Configure Your Auto Scaling Group to Send Notifications

You can configure your Auto Scaling group to send notifications to Amazon SNS when a scaling event, such as launching instances or terminating instances, takes place. Amazon SNS sends a notification with information about the instances to the email address that you specified.

To configure Amazon SNS notifications for your Auto Scaling group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Notifications** tab, choose **Create notification**.
5. On the **Create notifications** pane, do the following:

    a. For **Send a notification to:**, select your SNS topic.
    b. For **Whenever instances**, select the events to send the notifications for.
    c. Choose **Save**.

To configure Amazon SNS notifications for your Auto Scaling group using the AWS CLI

Use the following put-notification-configuration command:

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-
asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH"
 "autoscaling:EC2_INSTANCE_TERMINATE"
```

Test the Notification Configuration

To generate a notification for a launch event, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. Auto Scaling launches the EC2 instance, and you'll receive an email notification within a few minutes.

To change the desired capacity using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Details** tab, choose **Edit**.
5. For **Desired**, increase the current value by 1. Note that if this value exceeds **Max**, you must also increase the value of **Max** by 1.
6. Choose **Save**.
7. After a few minutes, you'll receive a notification email for the launch event. If you do not need the additional instance that you launched for this test, you can decrease **Desired** by 1. After a few minutes, you'll receive a notification email for the terminate event.

To change the desired capacity using the AWS CLI

Use the following set-desired-capacity command:

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg --
desired-capacity 2
```

Verify That You Received Notification of the Scaling Event

Check your email for a message from Amazon SNS and open the email. After you receive notification of a scaling event for your Auto Scaling group, you can confirm the scaling event by looking at the description of your Auto Scaling group. You'll need information from the notification email, such as the ID of the instance that was launched or terminated.

To verify that your Auto Scaling group has launched new instance using the console

1. Select your Auto Scaling group.
2. On the **Activity History** tab, the **Status** column shows the current status of your instance. For example, if the notification indicates that an instance has launched, use the refresh button to verify that the status of the launch activity is **Successful**.
3. On the **Instances** tab, you can view the current **Lifecycle** state of the instance whose ID you received in the notification email. After a new instance starts, its lifecycle state changes to `InService`.

To verify that your Auto Scaling group has launched new instance using the AWS CLI

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

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

The following example output shows that the group has two instances. Check for the instance whose ID you received in the notification email.

```
{
    "AutoScalingGroups": [
        {
```

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

Delete the Notification Configuration

You can delete your Auto Scaling notification configuration at any time.

To delete Auto Scaling notification configuration using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
3. Select your Auto Scaling group.
4. On the **Notifications** tab, choose **Delete** next to the notification.

To delete Auto Scaling notification configuration using the AWS CLI

Use the following `delete-notification-configuration` command:

```
aws autoscaling delete-notification-configuration --auto-scaling-group-
name my-asg --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

For information about deleting the Amazon SNS topic associated with your Auto Scaling group, and also deleting all the subscriptions to that topic, see Clean Up in the *Amazon Simple Notification Service Developer Guide*.

Logging Auto Scaling API Calls By Using AWS CloudTrail

Auto Scaling is integrated with CloudTrail, a service that captures API calls made by or on behalf of Auto Scaling in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the Auto Scaling console or from the Auto Scaling API. Using the information collected by CloudTrail, you can determine what request was made to Auto Scaling, the source IP address from which the request was made, who made the request, when it was made, and so on. For more information about CloudTrail, including how to configure and enable it, see the *AWS CloudTrail User Guide*.

Auto Scaling Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to Auto Scaling actions are tracked in log files. Auto Scaling records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the Auto Scaling actions are logged and are documented in the Auto Scaling API Reference. For example, calls to the **CreateLaunchConfiguration**, **DescribeAutoScalingGroup**, and **UpdateAutoScalingGroup** actions generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with account or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see **userIdentity** in the CloudTrail Event Reference section in the *AWS CloudTrail User Guide*.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see Configuring Amazon SNS Notifications in the *AWS CloudTrail User Guide*.

You can also aggregate Auto Scaling log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket in the *AWS CloudTrail User Guide*.

Understanding Auto Scaling Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that demonstrates the **CreateLaunchConfiguration** action.

```
{
    "Records": [
    {
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/iamUser1",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "iamUser1"
            },
        "eventTime": "2014-06-24T16:53:14Z",
        "eventSource": "autoscaling.amazonaws.com",
        "eventName": "CreateLaunchConfiguration",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
        "requestParameters": {
            "imageId": "ami-2f726546",
            "instanceType": "m1.small",
            "launchConfigurationName": "launch_configuration_1"
            },
        "responseElements": null,
        "requestID": "07a1becf-fbc0-11e3-bfd8-a5209058e7bb",
        "eventID": "ad30abf7-57db-4a6d-93fa-13deb1fd4cff"
        },
        ...additional entries
    ]
}
```

The following example shows a CloudTrail log entry that demonstrates the **DescribeAutoScalingGroups** action.

```
{
    "Records": [
    {
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/iamUser1",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "iamUser1"
            },
        "eventTime": "2014-06-23T23:20:56Z",
        "eventSource": "autoscaling.amazonaws.com",
        "eventName": "DescribeAutoScalingGroups",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
        "requestParameters": {
```

```
                "maxRecords": 20
                },
            "responseElements": null,
            "requestID": "0737e2ea-fb2d-11e3-bfd8-a5209058e7bb",
            "eventID": "0353fb04-281e-47d9-93bb-588bf2256538"
            },
            ...additional entries
        ]
}
```

The following example shows a CloudTrail log entry that demonstrates the
UpdateAutoScalingGroups action.

```
{
    "Records": [
    {
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/iamUser1",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "iamUser1"
            },
        "eventTime": "2014-06-24T16:54:46Z",
        "eventSource": "autoscaling.amazonaws.com",
        "eventName": "UpdateAutoScalingGroup",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Amazon CLI/AutoScaling 1.0.61.3 API 2011-01-01",
        "requestParameters": {
            "maxSize": 8,
            "minSize": 1,
            "autoScalingGroupName": "asg1"
            },
        "responseElements": null,
        "requestID": "3ed07c03-fbc0-11e3-bfd8-a5209058e7bb",
        "eventID": "b52ca0aa-5199-4873-a546-55f7c896a4ce"
        },
        ...additional entries
    ]

}
```

Controlling Access to Your Auto Scaling Resources

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

- Create users and groups under your organization's AWS account
- Assign unique security credentials to each user under your AWS account
- Control each user's permissions to perform tasks using AWS resources
- Allow the users in another AWS account to share your AWS resources
- Create roles for your AWS account and define the users or services that can assume them
- Use existing identities for your enterprise to grant permissions to perform tasks using AWS resources

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

For more information, see Identity and Access Management (IAM) or the IAM User Guide.

Contents

Auto Scaling Actions

In an IAM policy, you can specify any and all Auto Scaling actions. For Auto Scaling, use the following prefix with the name of the action: `autoscaling:`. For example:

`autoscaling:CreateAutoScalingGroup` and `autoscaling:CreateLaunchConfiguration`. You can also use wildcards. For example, use `autoscaling:*` to indicate all Auto Scaling actions.

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

Auto Scaling Resources

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

Auto Scaling Keys

For a list of context keys supported by each AWS service and a list of AWS-wide policy keys, see AWS Service Actions and Condition Context Keys and Available Keys for Conditions in the *IAM User Guide*.

Predefined AWS Managed Policies

The managed policies created by AWS grant the required permissions for common use cases. You can attach these policies to your IAM users. The following are the AWS managed policies for Auto Scaling.

- **AutoScalingConsoleFullAccess** — Grants access to all API actions used by the console for Auto Scaling resources. This includes all API actions for Auto Scaling, and selected API actions for Amazon EC2, CloudWatch, Elastic Load Balancing, and Amazon SNS.
- **AutoScalingConsoleReadOnlyAccess** — Grants access to the read-only API actions used by the console for Auto Scaling resources. This includes all read-only API actions for Auto Scaling, and selected read-only API actions for Amazon EC2, CloudWatch, Elastic Load Balancing, and Amazon SNS
- **AutoScalingFullAccess** — Grants access to all Auto Scaling API actions.
- **AutoScalingReadOnlyAccess** — Grants access to the read-only Auto Scaling API actions.

Customer Managed Policies

You can create custom IAM policies that grant your IAM users permissions to perform specific actions on specific resources. The following are example policies for Auto Scaling. Note that the resource is always "*", because you can't specify a particular Auto Scaling resource in a policy.

Example 1: Create and manage Auto Scaling launch configurations

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

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

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
```

```
      "Action": "autoscaling:*LaunchConfiguration*",
      "Resource": "*"
      }
   ]
}
```

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

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

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

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

Example 3: Change the capacity of Auto Scaling groups.

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

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

Launch Auto Scaling Instances with an IAM Role

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

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

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

For more information about IAM roles and instance profiles, see IAM Roles in the *IAM User Guide*.

Prerequisites

Create an IAM role for your EC2 instances. The console creates an instance profile with the same name as the IAM role. For more information, see Creating an IAM Role Using the Console in the *Amazon EC2 User Guide for Linux Instances.*

Create a Launch Configuration

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

For example, use the following create-launch-configuration command:

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

Create an Auto Scaling Group

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

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

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-
with-instance-profile --launch-configuration-name my-lc-with-instance-profile
 --availability-zones "us-west-2c" --max-size 1 --min-size 1
```

Troubleshooting Auto Scaling

Amazon Web Services provides specific and descriptive errors to help you troubleshoot Auto Scaling problems. You can find the error messages in the description of the Auto Scaling activities.

Contents

Retrieving an Error Message

To retrieve an error message from the description of Auto Scaling activities, use the describe-scaling-activities command as follows:

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

The following is an example response, where `StatusCode` contains the current status of the activity and `StatusMessage` contains the error message:

```
{
    "Activities": [
        {
            "Description": "Launching a new EC2 instance: i-4ba0837f",
            "AutoScalingGroupName": "my-asg",
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Details": "{"Availability Zone":"us-west-2c"}",
            "StartTime": "2013-08-19T20:53:29.930Z",
```

```
            "Progress": 100,
            "EndTime": "2013-08-19T20:54:02Z",
            "Cause": "At 2013-08-19T20:53:25Z a user request created an
 AutoScalingGroup...",
            "StatusCode": "Failed",
            "StatusMessage": "The image id 'ami-4edb0327' does not exist.
 Launching EC2 instance failed."
        }
    ]
}
```

The following tables list the types of error messages and provide links to the troubleshooting resources that you can use to troubleshoot your Auto Scaling issues.

EC2 Instance Launch Failures

Issue	Error Message
Auto Scaling group	AutoScalingGroup <Auto Scaling group name> not found. (p. 132)
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request ...... (p. 132)
AWS account	You are not subscribed to this service. Please see http://aws.amazon.com. (p. 132)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 132)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 133)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 133)
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>).... (p. 132)
Key pair	The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 131)
Launch configuration	The requested configuration is currently not supported. (p. 131)
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 133)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 131)

AMI Issues

Issue	Error Message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 134)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 134)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 135)

Issue	Error Message
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. 135)
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. 134)

Load Balancer Issues

Issue	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 136)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 136)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 136)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 136)

Capacity Limits

Issue	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 137)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>).... (p. 137)

Troubleshooting Auto Scaling: EC2 Instance Launch Failures

This page provides information about your EC2 instances that fail to launch with Auto Scaling, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see Retrieving an Error Message (p. 128).

When your EC2 instances fail to launch, you might get one or more of the following error messages:

Error Messages

- The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 131)
- The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 131)
- The requested configuration is currently not supported. (p. 131)

Auto Scaling User Guide
The security group <name of the security group>
does not exist. Launching EC2 instance failed.

The security group <name of the security group> does not exist. Launching EC2 instance failed.

- **Cause**: The security group specified in your launch configuration might have been deleted.
- **Solution**:
  1. Use the describe-security-groups command to get the list of the security groups associated with your account.
  2. From the list, select the security groups to use. To create a security group instead, use the create-security-group command.
  3. Create a new launch configuration.
  4. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed.

- **Cause**: The key pair that was used when launching the instance might have been deleted.
- **Solution**:
  1. Use the describe-key-pairs command to get the list of the key pairs available to you.
  2. From the list, select the key pair to use. To create a key pair instead, use the create-key-pair command.
  3. Create a new launch configuration.
  4. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

The requested configuration is currently not supported.

- **Cause**: Some options in your launch configuration might not be currently supported.
- **Solution**:
  1. Create a new launch configuration.

2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

AutoScalingGroup <Auto Scaling group name> not found.

- **Cause**: The Auto Scaling group might have been deleted.
- **Solution**: Create a new Auto Scaling group.

The requested Availability Zone is no longer supported. Please retry your request ......

- **Error Message**: The requested Availability Zone is no longer supported. Please retry your request by not specifying an Availability Zone or choosing <list of available Availability Zones>. Launching EC2 instance failed.
- **Cause**: The Availability Zone associated with your Auto Scaling group might not be currently available.
- **Solution**: Update your Auto Scaling group with the recommendations in the error message.

Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)....

- **Error Message**: Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>). Please retry your request by not specifying an Availability Zone or choosing <list of Availability Zones that supports the instance type>. Launching EC2 instance failed.
- **Cause**: The instance type associated with your launch configuration might not be currently available in the Availability Zones specified in your Auto Scaling group.
- **Solution**: Update your Auto Scaling group with the recommendations in the error message.

You are not subscribed to this service. Please see http://aws.amazon.com.

- **Cause**: Your AWS account might have expired.
- **Solution**: Go to http://aws.amazon.com and choose **Sign Up Now** to open a new account.

Invalid device name upload. Launching EC2 instance failed.

- **Cause**: The block device mappings in your launch configuration might contain block device names that are not available or currently not supported.
- **Solution**:

Auto Scaling User Guide
Value (<name associated with the instance storage
device>) for parameter virtualName is invalid...

1. Use the describe-volumes command to see how the volumes are exposed to the instance.

2. Create a new launch configuration using the device name listed in the volume description.

3. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

Value (<name associated with the instance storage device>) for parameter virtualName is invalid...

- **Error Message**: Value (<name associated with the instance storage device>) for parameter virtualName is invalid. Expected format: 'ephemeralNUMBER'. Launching EC2 instance failed.

- **Cause**: The format specified for the virtual name associated with the block device is incorrect.

- **Solution**:

  1. Create a new launch configuration by specifying the device name in the `virtualName` parameter. For information about the device name format, see Instance Store Device Names in the *Amazon EC2 User Guide for Linux Instances.*

  2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

EBS block device mappings not supported for instance-store AMIs.

- **Cause**: The block device mappings specified in the launch configuration are not supported on your instance.

- **Solution**:

  1. Create a new launch configuration with block device mappings supported by your instance type. For more information, see Block Device Mapping in the *Amazon EC2 User Guide for Linux Instances*.

  2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed.

- **Cause**: Your cluster placement group contains an invalid instance type.

- **Solution**:

  1. For information about valid instance types supported by the placement groups, see Placement Groups in the *Amazon EC2 User Guide for Linux Instances*.

  2. Follow the instructions detailed in the Placement Groups to create a new placement group.

  3. Alternatively, create a new launch configuration with the supported instance type.

  4. Update your Auto Scaling group with new placement group or launch configuration using the update-auto-scaling-group command.

Troubleshooting Auto Scaling: AMI Issues

This page provides information about the issues associated with your AMIs, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see Retrieving an Error Message (p. 128).

When your EC2 instances fail to launch due to issues with your AMI, you might get one or more of the following error messages.

Error Messages
- The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 134)
- AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 134)
- Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed. (p. 134)
- Value (<ami ID>) for parameter virtualName is invalid. (p. 135)
- 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. 135)

The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed.

- **Cause**: The AMI might have been deleted after creating the launch configuration.
- **Solution**:
  1. Create a new launch configuration using a valid AMI.
  2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed.

- **Cause**: You might have just created your AMI (by taking a snapshot of a running instance or any other way), and it might not be available yet.
- **Solution**: You must wait for your AMI to be available and then create your launch configuration.

Non-Windows AMIs with a virtualization type of 'hvm' currently may only be used with Cluster Compute instance types. Launching EC2 instance failed.

- **Cause**: The Linux AMI with hvm virtualization cannot be used to launch a non-cluster compute instance.
- **Solution**:
  1. Create a new launch configuration using an AMI with a virtualization type of paravirtual to launch a non-cluster compute instance.
  2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

Value (<ami ID>) for parameter virtualName is invalid.

- **Cause**: Incorrect value. The `virtualName` parameter refers to the virtual name associated with the device.
- **Solution**:
    1. Create a new launch configuration by specifying the name of the virtual device of your instance for the `virtualName` parameter.
    2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

The requested instance type's architecture (i386) does not match the architecture in the manifest for ami-6622f00f (x86_64). Launching ec2 instance failed.

- **Cause**: The architecture of the `InstanceType` mentioned in your launch configuration does not match the image architecture.
- **Solution**:
    1. Create a new launch configuration using the AMI architecture that matches the architecture of the requested instance type.
    2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

Troubleshooting Auto Scaling: Load Balancer Issues

This page provides information about issues caused by the load balancer associated with your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see Retrieving an Error Message (p. 128).

When your EC2 instances fail to launch due to issues with the load balancer associated with your Auto Scaling group, you might get one or more of the following error messages.

Error Messages

- Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 136)
- There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 136)
- EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 136)
- EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed. (p. 136)
- The security token included in the request is invalid. Validating load balancer configuration failed. (p. 136)

Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.

- **Cause 1**: The load balancer has been deleted.
- **Solution 1**:

  1. Check to see if your load balancer still exists. You can use the describe-load-balancers command.

  2. If you see your load balancer listed in the response, see **Cause 2**.

  3. If you do not see your load balancer listed in the response, you can either create a new load balancer and then create a new Auto Scaling group or you can create a new Auto Scaling group without the load balancer.

- **Cause 2**: The load balancer name was not specified in the right order when creating the Auto Scaling group.
- **Solution 2**: Create a new Auto Scaling group and specify the load balancer name at the end.

There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.

- **Cause**: The specified load balancer might have been deleted.
- **Solution**: You can either create a new load balancer and then create a new Auto Scaling group or create a new Auto Scaling group without the load balancer.

EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.

- **Cause**: The specified instance does not exist in the VPC.
- **Solution**: You can either delete your load balancer associated with the instance or create a new Auto Scaling group.

EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed.

- **Cause**: The load balancer is in EC2-Classic but the Auto Scaling group is in a VPC.
- **Solution**: Ensure that the load balancer and the Auto Scaling group are in the same network (EC2-Classic or a VPC).

The security token included in the request is invalid. Validating load balancer configuration failed.

- **Cause**: Your AWS account might have expired.

- **Solution**: Check whether your AWS account is valid. Go to http://aws.amazon.com and choose **Sign Up Now** to open a new account.

Troubleshooting Auto Scaling: Capacity Limits

This page provides information about issues with the capacity limits of your Auto Scaling group, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see Retrieving an Error Message (p. 128).

If your EC2 instances fail to launch due to issues with the capacity limits of your Auto Scaling group, you might get one or more of the following error messages.

Error Messages
- We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>).... (p. 137)
- <number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 137)

We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)....

- **Error Message**: We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>). Our system will be working on provisioning additional capacity. You can currently get <instance type> capacity by not specifying an Availability Zone in your request or choosing <list of Availability Zones that currently supports the instance type>. Launching EC2 instance failed.
- **Cause**: At this time, Auto Scaling cannot support your instance type in your requested Availability Zone.
- **Solution**:
  1. Create a new launch configuration by following the recommendations in the error message.
  2. Update your Auto Scaling group with the new launch configuration using the update-auto-scaling-group command.

<number of instances> instance(s) are already running. Launching EC2 instance failed.

- **Cause**: The Auto Scaling group has reached the limit set by the `DesiredCapacity` parameter.
- **Solution**:
  - Update your Auto Scaling group by providing a new value for the `--desired-capacity` parameter using the update-auto-scaling-group command.
  - If you've reached your limit for number of EC2 instances, you can request an increase. For more information, see AWS Service Limits.

Auto Scaling Resources

The following related resources can help you as you work with this service.

- **Auto Scaling** – The primary web page for information about Auto Scaling.
- **Auto Scaling Technical FAQ** – The answers to questions customers ask about Auto Scaling.
- **Amazon EC2 Discussion Forum** – Get help from the community.

- **Classes & Workshops** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- **AWS Whitepapers** – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Document History

The following table describes important additions to the Auto Scaling documentation.

Feature	Description	Release Date
Monitoring improvements	Auto Scaling group metrics no longer require that you enable detailed monitoring. You can now enable group metrics collection and view metrics graphs from the **Monitoring** tab in the console. For more information, see Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch (p. 105).	18 August 2016
Support for Application Load Balancers	Attach one or more target groups to a new or existing Auto Scaling group. For more information, see Attaching a Load Balancer to Your Auto Scaling Group (p. 47).	11 August 2016
Events for lifecycle hooks	Auto Scaling sends events to CloudWatch Events when it executes lifecycle hooks. For more information, see Getting CloudWatch Events When Your Auto Scaling Group Scales (p. 110).	24 February 2016
Instance protection	Prevent Auto Scaling from selecting specific instances for termination when scaling in. For more information, see Instance Protection (p. 87).	07 December 2015
Step scaling policies	Create a scaling policy that enables you to scale based on the size of the alarm breach. For more information, see Scaling Policy Types (p. 71).	06 July 2015
Update load balancer	Attach a load balancer to or detach a load balancer from an existing Auto Scaling group. For more information, see Attaching a Load Balancer to Your Auto Scaling Group (p. 47).	11 June 2015
Support for ClassicLink	Link EC2-Classic instances in your Auto Scaling group to a VPC, enabling communication between these linked EC2-Classic instances and instances in the VPC using private IP addresses. For more information, see Linking EC2-Classic Instances to a VPC (p. 28).	19 January 2015
Lifecycle hooks	Hold your newly launched or terminating instances in a pending state while you perform actions on them. For more information, see Auto Scaling Lifecycle Hooks (p. 89).	30 July 2014

Feature	Description	Release Date
Detach instances	Detach instances from an Auto Scaling group. For more information, see Detach EC2 Instances From Your Auto Scaling Group (p. 65).	30 July 2014
Put instances into a Standby state	Put instances that are in an `InService` state into a `Standby` state. For more information, see Temporarily Removing Instances from Your Auto Scaling Group (p. 95).	30 July 2014
Manage tags	Manage your Auto Scaling groups using the AWS Management Console. For more information, see Tagging Auto Scaling Groups and Instances (p. 43).	01 May 2014
Support for Dedicated Instances	Launch Dedicated Instances by specifying a placement tenancy attribute when you create a launch configuration. For more information, see Instance Placement Tenancy (p. 27).	23 April 2014
Create a group or launch configuration from an EC2 instance	Create an Auto Scaling group or a launch configuration using an EC2 instance. For information about creating a launch configuration using an EC2 instance, see Creating a Launch Configuration Using an EC2 Instance (p. 22) For information about creating an Auto Scaling group using an EC2 instance, see Creating an Auto Scaling Group Using an EC2 Instance (p. 40).	02 January 2014
Attach instances	Enable Auto Scaling for an EC2 instance by attaching the instance to an existing Auto Scaling group. For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 61).	02 January 2014
View account limits	View the limits on Auto Scaling resources for your account. For more information, see Auto Scaling Limits (p. 9).	02 January 2014
Console support for Auto Scaling	Access Auto Scaling using the AWS Management Console. For more information, see Getting Started with Auto Scaling (p. 12).	10 December 2013
Assign a public IP address	Assign a public IP address to an instance launched into a VPC. For more information, see Launching Auto Scaling Instances in a VPC (p. 26).	19 September 2013
Instance termination policy	Specify an instance termination policy for Auto Scaling to use when terminating EC2 instances. For more information , see Controlling Which Instances Auto Scaling Terminates During Scale In (p. 84).	17 September 2012
Support for IAM roles	Launch EC2 instances with an IAM instance profile. You can use this feature to assign IAM roles to your instances, allowing your applications to access other AWS services securely. For more information , see Launch Auto Scaling Instances with an IAM Role (p. 126).	11 June 2012
Support for Spot Instances	Request Spot Instances in Auto Scaling groups by specifying a Spot Instance bid price in your launch configuration. For more information, see Launching Spot Instances in Your Auto Scaling Group (p. 30).	7 June 2012

Feature	Description	Release Date
Tag groups and instances	Tag Auto Scaling groups and specify that the tag also applies to EC2 instances launched after the tag was created. For more information, see Tagging Auto Scaling Groups and Instances (p. 43).	26 January 2012
Support for Amazon SNS	Use Amazon SNS to receive notifications whenever Auto Scaling launches or terminates EC2 instances. For more information, see Getting SNS Notifications When Your Auto Scaling Group Scales (p. 116). 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 out without adding the launched instance to the load balancer (LoadBalancer). For more information, see the ProcessType API data type. • The ForceDelete flag in the DeleteAutoScalingGroup command that tells Auto Scaling to delete the Auto Scaling group with the instances associated to it without waiting for the instances to be terminated first. For more information, see the DeleteAutoScalingGroup API command.	20 July 2011
Scheduled scaling actions	You can now create scheduled scaling actions. For more information, see Scheduled Scaling (p. 67).	2 December 2010
Support for Amazon VPC	Added support for Amazon VPC. For more information, see Launching Auto Scaling Instances in a VPC (p. 26).	2 December 2010
Support for HPC clusters	Added support for high performance computing (HPC) clusters.	2 December 2010
Support for health checks	Added support for using Elastic Load Balancing health checks with Auto Scaling-managed EC2 instances. For more information, see Adding Health Checks to Your Auto Scaling Group (p. 49).	2 December 2010
Support for CloudWatch alarms	Removed the older trigger mechanism and redesigned Auto Scaling to use the CloudWatch alarm feature. For more information, see Dynamic Scaling (p. 70).	2 December 2010
Suspend and resume scaling	You can now suspend and resume scaling processes.	2 December 2010
Support for IAM	Auto Scaling now supports IAM. For more information, see Controlling Access to Your Auto Scaling Resources (p. 124).	2 December 2010