Amazon EC2 Auto Scaling

User Guide

aws

Amazon EC2 Auto Scaling: User Guide

Table of Contents

What Is Amazon EC2 Auto Scaling?

Amazon EC2 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 Amazon EC2 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 Amazon EC2 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, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 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 one instance, a desired capacity of two instances, and a maximum size of four 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 Amazon EC2 Auto Scaling, see Benefits of Auto Scaling (p. 3).

Auto Scaling Components

The following table describes the key components of Amazon EC2 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. 47).
	Configuration templates Your group uses a *launch template* or a *launch configuration* as a configuration template for its EC2 instances. 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 Templates (p. 27) and Launch Configurations (p. 36).

	Scaling options
	Amazon EC2 Auto Scaling provides several ways for you to scale your Auto Scaling groups. For example, you can configure a group to scale based on the occurrence of specified conditions (dynamic scaling) or on a schedule. For more information, see Scaling Options (p. 94).

Getting Started

If you're new to Amazon EC2 Auto Scaling, we recommend that you review Auto Scaling Lifecycle (p. 7) before you begin.

To begin, complete the Getting Started with Amazon EC2 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 Amazon EC2 Auto Scaling

If you've signed up for an AWS account, you can access Amazon EC2 Auto Scaling by signing into the AWS Management Console, choosing **EC2** from the console home page, and then choosing **Auto Scaling Groups** from the navigation pane.

You can also access Amazon EC2 Auto Scaling using the Amazon EC2 Auto Scaling API. Amazon EC2 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 Amazon EC2 Auto Scaling, see Actions in the *Amazon EC2 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.

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, macOS, and Linux. To get started, see AWS Command Line Interface User Guide. For more information, see autoscaling in the *AWS CLI Command 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, see the AWS Tools for PowerShell Cmdlet Reference.

For information about your credentials for accessing AWS, see AWS Security Credentials in the *Amazon Web Services General Reference*. For information about regions and endpoints for calls to Amazon EC2 Auto Scaling, see Amazon EC2 Auto Scaling Regions and Endpoints in the *AWS General Reference*.

Pricing for Amazon EC2 Auto Scaling

There are no additional fees with Amazon EC2 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 configure automatic scaling for all of the scalable AWS resources for your application, use AWS Auto Scaling. With AWS Auto Scaling, you can also simplify the process of defining dynamic scaling policies for your Auto Scaling groups and use predictive scaling to scale your Amazon EC2 capacity in advance of predicted traffic changes. For more information, see the AWS Auto Scaling User Guide.

To automatically distribute incoming application traffic across multiple instances in your Auto Scaling group, use Elastic Load Balancing. For more information, see the 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 Amazon EC2 Auto Scaling API for your account, use AWS CloudTrail. The data logged includes calls made by the AWS Management Console, command line tools, and other services. For more information, see the AWS CloudTrail User Guide.

Benefits of Auto Scaling

Adding Amazon EC2 Auto Scaling to your application architecture is one way to maximize the benefits of the AWS Cloud. When you use Amazon EC2 Auto Scaling, your applications gain the following benefits:

- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 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 needed and terminating them when they aren't.
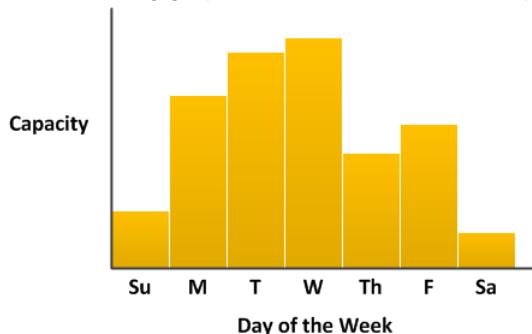
Contents

Example: Covering Variable Demand

To demonstrate some of the benefits of Amazon EC2 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 demand on the application increases significantly.
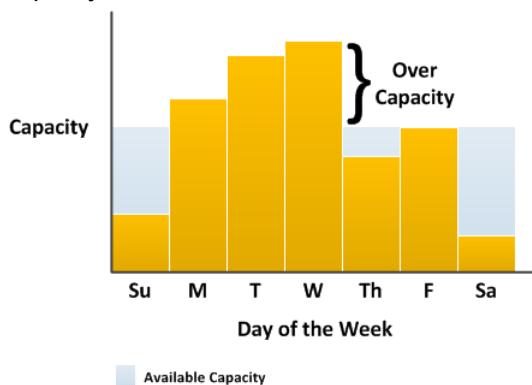
The following graph shows how much of the application's capacity is used over the course of a week.
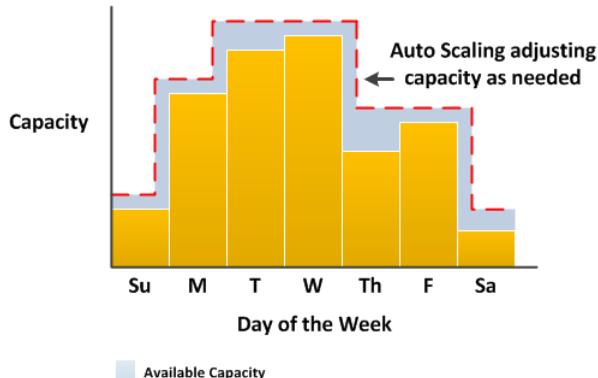
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity. 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 demand 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 demand on the application exceeds its capacity.
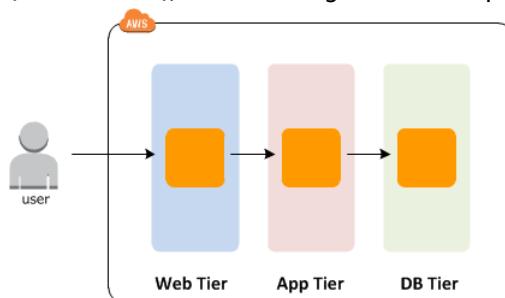
By adding Amazon EC2 Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. Because Amazon EC2 Auto Scaling uses EC2 instances, you only have to pay for the instances you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while minimizing expenses.
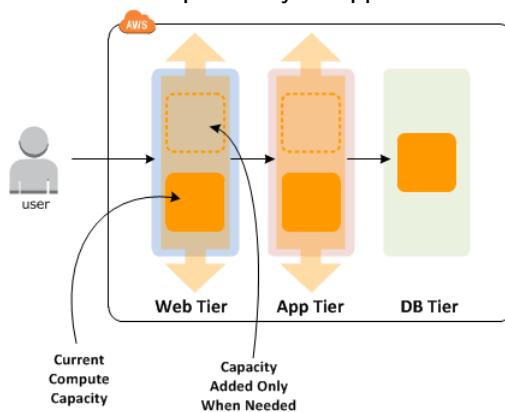
Example: Web App Architecture

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.

Amazon EC2 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 helps 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 Elastic Load Balancing (p. 76).

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*, which are engineered to be isolated from failures in other Availability Zones. They provide inexpensive, low-latency network connectivity to other Availability Zones in the same Region. For more information, see Amazon EC2 Auto Scaling Regions and Endpoints in the *Amazon Web Services General Reference*.
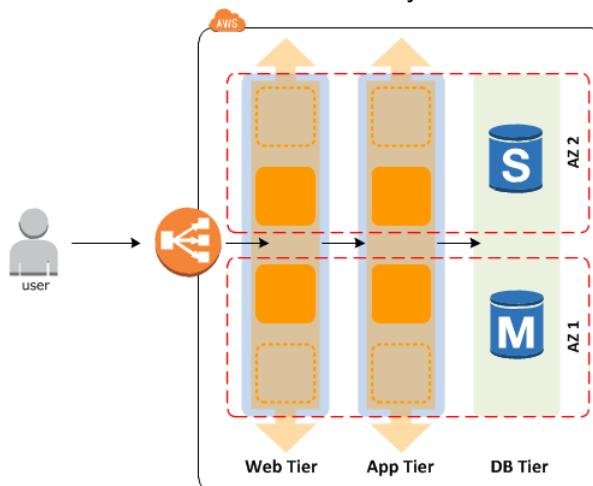
Amazon EC2 Auto Scaling enables you to take advantage of the safety and reliability of geographic redundancy by spanning Auto Scaling groups across multiple Availability Zones within a Region. When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected 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 select the subnets for your EC2 instances when you create or update the Auto Scaling group. You can select one or more subnets per Availability Zone. For more information, see VPCs and Subnets in the *Amazon VPC User Guide*.

Instance Distribution

Amazon EC2 Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group. Amazon EC2 Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Amazon EC2 Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds. For Auto Scaling groups in a VPC, if there are multiple subnets in an Availability Zone, Amazon EC2 Auto Scaling selects a subnet from the Availability Zone at random.

Rebalancing Activities

After certain actions occur, your Auto Scaling group can become unbalanced between Availability Zones. Amazon EC2 Auto Scaling compensates by rebalancing the Availability Zones. The following actions can lead to rebalancing activity:

- You change the Availability Zones for your group.
- You explicitly terminate or detach instances and the group becomes unbalanced.
- An Availability Zone that previously had insufficient capacity recovers and has additional capacity available.
- An Availability Zone that previously had a Spot price above your maximum price now has a Spot price below your maximum price.

When rebalancing, Amazon EC2 Auto Scaling launches new instances before terminating the old ones, so that rebalancing does not compromise the performance or availability of your application.

Because Amazon EC2 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.
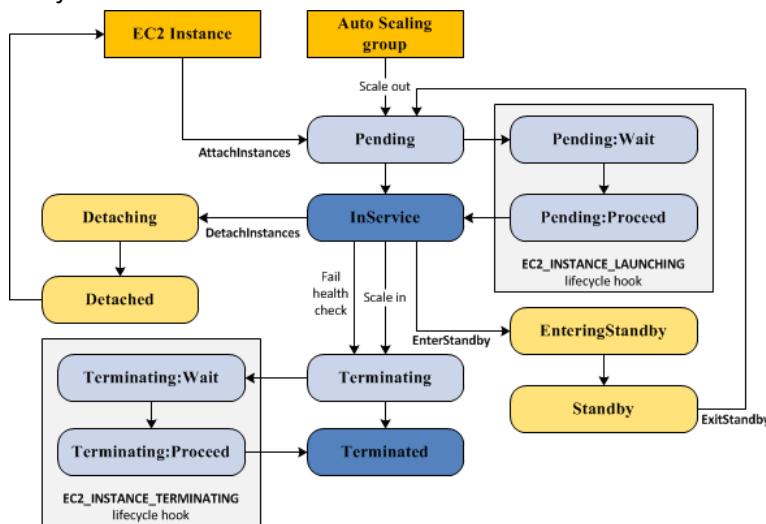
Auto Scaling Lifecycle

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 Amazon EC2 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:

- You manually increase the size of the group. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).
- You create a scaling policy to automatically increase the size of the group based on a specified increase in demand. For more information, see Dynamic Scaling for Amazon EC2 Auto Scaling (p. 108).
- You set up scaling by schedule to increase the size of the group at a specific time. For more information, see Scheduled Scaling for Amazon EC2 Auto Scaling (p. 105).

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:

- A scale-in event occurs, and Amazon EC2 Auto Scaling chooses to terminate this instance in order to reduce the size of the Auto Scaling group. For more information, see Controlling Which Auto Scaling Instances Terminate During Scale In (p. 137).
- You put the instance into a `Standby` state. For more information, see Enter and Exit Standby (p. 9).
- You detach the instance from the Auto Scaling group. For more information, see Detach an Instance (p. 9).
- The instance fails a required number of health checks, so it is removed from the Auto Scaling group, terminated, and replaced. For more information, see Health Checks for Auto Scaling Instances (p. 160).

Scale In

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. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).
- You create a scaling policy to automatically decrease the size of the group based on a specified decrease in demand. For more information, see Dynamic Scaling for Amazon EC2 Auto Scaling (p. 108).
- You set up scaling by schedule to decrease the size of the group at a specific time. For more information, see Scheduled Scaling for Amazon EC2 Auto Scaling (p. 105).

It is important that you create a corresponding 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.

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. 98).

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. 102).

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 Amazon EC2 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 are fully configured, they are attached to the Auto Scaling group and they enter the `InService` state.

When Amazon EC2 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 Amazon EC2 Auto Scaling Lifecycle Hooks (p. 142).

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. 150).

Amazon EC2 Auto Scaling Service Quotas

Your AWS account has the following default quotas, formerly referred to as limits, for Amazon EC2 Auto Scaling.

Default Quotas

- Launch configurations per Region: 200
- Auto Scaling groups per Region: 200

To view the current quotas for your account, open the Amazon EC2 console at https://console.aws.amazon.com/ec2/ and navigate to the **Limits** page. You can also use the describe-account-limits command. To request an increase, use the Auto Scaling Limits form.

Auto Scaling Group Quotas

- 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

Scaling Policy Quotas

- Step adjustments per scaling policy: 20

Auto Scaling API Quotas

- You can use `AttachInstances`, `DetachInstances`, `EnterStandby`, and `ExitStandby` with at most 20 instance IDs at a time.
- You can use `AttachLoadBalancers` and `DetachLoadBalancers` with at most 10 load balancers at a time.
- You can use `AttachLoadBalancerTargetGroups` and `DetachLoadBalancerTargetGroups` with at most 10 target groups at a time.

For information about the service quotas for other services, see Service Endpoints and Quotas in the *Amazon Web Services General Reference*.

Setting Up Amazon EC2 Auto Scaling

Before you start using Amazon EC2 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 Amazon EC2 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://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

AWS sends you a confirmation email 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*.

Getting Started with Amazon EC2 Auto Scaling

When you use Amazon EC2 Auto Scaling, you must use certain building blocks to get started. This tutorial walks you through the process for setting up building blocks to create a basic infrastructure for Amazon EC2 Auto Scaling.

Before you create an Auto Scaling group for use with your application, review your application thoroughly as it runs in the AWS Cloud. Consider the following:

- How many Availability Zones the Auto Scaling group should span.
- What existing resources can be used, such as security groups or Amazon Machine Images (AMIs).
- Whether you want to scale to increase or decrease capacity, or if you just want to ensure that a specific number of servers are always running. Keep in mind that Amazon EC2 Auto Scaling can do both simultaneously.
- What metrics have the most relevance to your application's performance.
- How long it takes to launch and configure a server.

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

The following instructions:

- Create a configuration template that defines your EC2 instances. You can choose either the launch template or the launch configuration instructions, based on your preference.
- Create an Auto Scaling group to maintain a fixed number of instances even if an instance becomes unhealthy.
- Optionally, delete this basic infrastructure.

This tutorial assumes that you are familiar with launching EC2 instances and that you have already created a key pair and a security group. For more information, see Setting Up with Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*.

To get started, you can launch a single free tier eligible Linux instance. If you created your AWS account less than 12 months ago, and have not already exceeded the free tier benefits for Amazon EC2, it will not cost you anything to complete this tutorial, because we help you select options that are within the free tier benefits. Otherwise, when you follow this tutorial, you incur the standard Amazon EC2 usage fees from the time that the instance launches until you delete the Auto Scaling group (which is the final task of this tutorial) and the instance status changes to `terminated`.

Tasks

Step 1: Create a Launch Template

For this step, you create a launch template that specifies the type of EC2 instance that Amazon EC2 Auto Scaling creates for you. Include information such as the ID of the Amazon Machine Image (AMI) to use, the instance type, the key pair, and security groups.
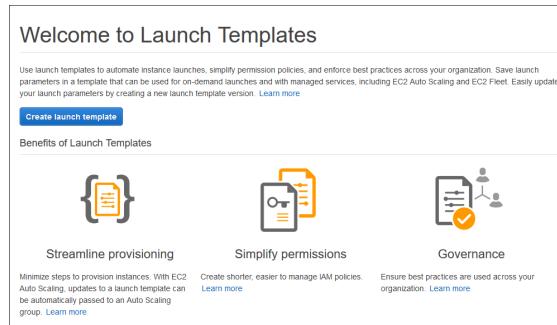
For detailed instructions for creating launch templates, see Creating a Launch Template for an Auto Scaling Group (p. 27).

> **Note**
> Alternatively, you can use a launch configuration to create an Auto Scaling group instead of using a launch template. For the launch configuration instructions, see Create a launch configuration.

To create a launch template

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 an AWS Region. The Amazon EC2 Auto Scaling resources that you create are tied to the Region that you specify.
3. On the navigation pane, choose **Launch Templates**.
4. Choose **Create launch template**.

5. Choose **Create a new template**. For **Launch template name**, enter a name (for example, `my_template`).
6. For **Template version description**, enter a description of the launch template to help you remember what this template is used for (for example, `test launch template for an Auto Scaling group`).
7. For **AMI ID**, choose a version of Amazon Linux 2 (HVM) from the **Quick Start** list. The Amazon Machine Image (AMI) serves as a basic configuration template for your instances.
8. For **Instance type**, choose a hardware configuration that is compatible with the AMI that you specified. Note that the free tier Linux server is a `t2.micro` instance.

   > **Note**
   > If your account is less than 12 months old, you can use a `t2.micro` instance for free within certain usage limits. For more information, see AWS Free Tier.

9. (Optional) For **Key pair name**, choose an existing key pair. You use key pairs to connect to an Amazon EC2 instance with SSH. Connecting to an instance is not included as part of this tutorial. Therefore, you don't need to specify a key pair unless you intend to connect to your instance.
10. Leave **Network type** set to **VPC**.
11. For **Security Groups**, specify the security group in the same VPC that you plan to use as the VPC for your Auto Scaling group. If you don't specify a security group, your instance is automatically associated with the default security group for the VPC.
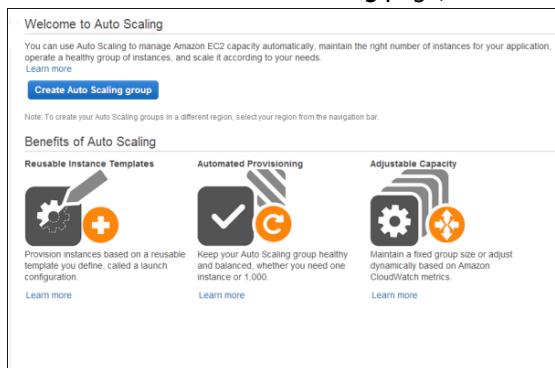
12. You can leave **Network Interfaces** empty. Leaving the setting empty creates a primary network interface with IP addresses that we select for your instance (based on the subnet to which the network interface is established). If instead you choose to specify a network interface, the security group must be a part of it.

13. Scroll down and choose **Create launch template**.

14. On the confirmation page, choose **Create Auto Scaling group**.

If you are not currently using launch templates and prefer not to create one now, you can create a launch configuration instead.

A launch configuration is similar to a launch template, in that it specifies the type of EC2 instance that Amazon EC2 Auto Scaling creates for you. You create the launch configuration by including information such as the ID of the Amazon Machine Image (AMI) to use, the instance type, the key pair, and security groups.

To create a launch configuration

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

2. On the navigation bar, select an AWS Region. The Auto Scaling resources that you create are tied to the Region that you specify.

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

4. On the **Welcome to Auto Scaling** page, choose **Create Auto Scaling group**.

5. On the **Create Auto Scaling Group** page, choose **Launch Configuration, Create a new launch configuration**, and then choose **Next Step**.

6. For the **Choose AMI** step, there is a list of basic configurations, called Amazon Machine Images (AMIs), that serve as templates for your instances. Choose **Select** for the Amazon Linux 2 AMI.

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

   Note
   If your account is less than 12 months old, you can use a `t2.micro` instance for free within certain usage limits. For more information, see AWS Free Tier.

8. For the **Configure details** step, do the following:

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

   b. For **Advanced Details**, choose an IP address type. To provide internet connectivity to instances in a VPC, choose an option that assigns a public IP address. If an instance is launched into a default VPC, the default is to assign a public IP address. If you want to provide internet connectivity to your instance but aren't sure whether you have a default VPC, choose **Assign a public IP address to every instance**.

   c. Choose **Skip to review**.

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

10. For the **Review** step, choose **Create launch configuration**.

11. Complete the **Select an existing key pair or create a new key pair** step as instructed. Connecting to an instance is not included 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**.

13. Choose **Create an Auto Scaling group using this launch configuration**. The wizard to create an Auto Scaling group is displayed.

Step 2: Create an Auto Scaling Group

An Auto Scaling group is a collection of EC2 instances, and is the core of Amazon EC2 Auto Scaling. When you create an Auto Scaling group, you include information such as the subnets for the instances and the initial number of instances to start with.

Use the following procedure to continue where you left off after creating either a launch template or a launch configuration.

> **Note**
> Amazon EC2 Auto Scaling has recently changed the Auto Scaling group interface. By default, you're shown the old user interface, but you can switch to the new user interface. This topic contains steps for both.
> To switch to the new user interface, on the **Auto Scaling Groups** page, on the banner at the top of the page, choose **Go to the new console**.

To create an Auto Scaling group (new console)

1. From the new console, on the **Auto Scaling groups** page, choose **Create an Auto Scaling group**.

2. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.

3. For **Launch template**, choose one of the following options:

   a. [Launch template only] Choose the launch template that you created and then choose the default version of the launch template to use when scaling out.

   b. [Launch configuration only] Choose **Switch to launch configuration** and choose the launch configuration that you created.

4. Choose **Next**.

   The **Configure settings** page appears, allowing you to configure network settings and giving you options for launching On-Demand and Spot Instances across multiple instance types (if you chose a launch template).

5. [Launch template only] Keep **Purchase options and instance types** set to **Adhere to the launch template** to quickly create and configure an Auto Scaling group.

6. Keep **Network** set to the default VPC for your chosen AWS Region, or select your own VPC. The default VPC is automatically configured to provide internet connectivity to your instance. This VPC includes a public subnet in each Availability Zone in the Region.

7. For **Subnet**, choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability.

8. Keep the rest of the defaults for this tutorial and choose **Skip to review**.

   > **Note**
   > The initial size of the group is determined by its desired capacity. The default value is 1 instance.

9. On the **Review** page, review the information for the group, and then choose **Create Auto Scaling group**.

We recommend the new user interface, but you can still temporarily use the old user interface.

To create an Auto Scaling group (old console)

1. For the **Configure Auto Scaling group details** step, do the following:

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

    b. [Launch template] For **Launch template version**, choose the default version of the launch template to use when scaling out.

    c. [Launch template] For **Fleet Composition**, choose **Adhere to the launch template**.

    d. Keep **Group size** set to the default value of 1 instance for this tutorial.

    e. Keep **Network** set to the default VPC for your chosen AWS Region, or select your own VPC. The default VPC is automatically configured to provide internet connectivity to your instance. This VPC includes a public subnet in each Availability Zone in the Region.

    f. For **Subnet**, choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability.

    g. 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 an Auto Scaling group, you are ready to verify that the group has launched an EC2 instance.

On the **Auto Scaling groups** page, complete the following procedure.

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

1. Select the check box next to the Auto Scaling group that you just created.

    The first tab available is the **Details** tab, showing information about the Auto Scaling group.

2. Choose the second tab, **Activity**. Under **Activity history**, you can view the progress of activities that are associated with the Auto Scaling group. The **Status** column shows the current status of your instance. While your instance is launching, the status column shows `PreInService`. 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. (Old console: The **Activity History** tab is where you can view the status of the instance. While your instance is launching, the status column shows `In progress`.)

3. On the **Instance management** tab, under **Instances**, you can view the status of the instance. (Old console: The **Instances** tab is where you can view the status of the instance.)

4. Verify that your instance launched successfully. It takes a short time for an instance to launch.

   The **Lifecycle** column shows the state of your instance. Verify 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.

(Optional) Terminate an Instance in Your Auto Scaling Group

You can use these steps to learn more about how Amazon EC2 Auto Scaling works, specifically, how it launches new instances when necessary. The minimum size for the Auto Scaling group that you created in this tutorial is one instance. Therefore, if you terminate that running instance, Amazon EC2 Auto Scaling must launch a new instance to replace it.

1. On the **Instance management** tab, under **Instances**, select the ID of the instance. (Old console: The **Instances** tab is where you can select the ID of the instance.)

   This takes you to the **Instances** page in the Amazon EC2 console, where you can terminate the instance.

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

3. On the navigation pane, choose **Auto Scaling Groups**. Select your Auto Scaling group and choose the **Activity** tab. (Old console: **Activity History** tab).

   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. In the activity history, when the scaling activity starts, you see an entry for the termination of the first instance and an entry for the launch of a new instance.

4. On the **Instance management** tab, the **Instances** section shows the new instance only. (Old console: The **Instances** tab is where you can see this.)

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

Step 4: Next Steps

Go to the next step if you would like to delete the basic infrastructure for automatic scaling that you just created. Otherwise, you can use this infrastructure as your base and try one or more of the following:

- Manually scale your Auto Scaling group. For more information, see Setting Capacity Limits (p. 95) and Manual Scaling (p. 96).

- Learn how to automatically scale in response to changes in resource utilization. If the load increases, your Auto Scaling group can scale out (add instances) to handle the demand. For more information, see Target Tracking Scaling Policies (p. 110).

- Configure an SNS notification to notify you whenever your Auto Scaling group scales. For more information, see Amazon SNS Notifications (p. 162).

Step 5: (Optional) Delete Your Scaling Infrastructure

You can either delete your scaling infrastructure or delete just your Auto Scaling group and keep your launch template or configuration to use later.

If you launched an instance that is not within the AWS Free Tier, you should terminate your instances to prevent additional charges. When you terminate the instance, the data associated with it will also be deleted.

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 the check box next to your Auto Scaling group.
4. Choose **Delete**. (Old console: Choose **Actions**, **Delete**.) When prompted for confirmation, choose **Delete**.

   A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. When the deletion has occurred, the **Desired**, **Min**, and **Max** columns show 0 instances for the Auto Scaling group. It takes a few minutes to terminate the instance and delete the group. Refresh the list to see the current state.

Skip the following procedure if you would like to keep your launch template.

To delete your launch template

1. On the navigation pane, choose **Launch Templates**.
2. Select your launch template.
3. Choose **Actions**, **Delete template**. When prompted for confirmation, choose **Delete launch template**.

Skip the following procedure if you would like to 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.
3. Choose **Actions**, **Delete launch configuration**. When prompted for confirmation, choose **Yes, Delete**.

Tutorial: Set Up a Scaled and Load-Balanced Application

Important
Before you explore this tutorial, we recommend that you first review the following introductory tutorial: Getting Started with Amazon EC2 Auto Scaling (p. 12).

Registering your Auto Scaling group with an Elastic Load Balancing load balancer helps you set up a load-balanced application. Elastic Load Balancing works with Amazon EC2 Auto Scaling to load balance incoming traffic across your healthy Amazon EC2 instances. This increases the scalability and availability of your application. You can enable Elastic Load Balancing within a single Availability Zone or multiple Availability Zones to increase the fault tolerance of your applications. For more information, see Elastic Load Balancing and Amazon EC2 Auto Scaling (p. 76).

In this tutorial, we cover the basics steps for setting up a load-balanced application when the Auto Scaling group is created. To register an existing Auto Scaling group with a load balancer, see Attaching a Load Balancer (p. 77) instead.

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. We recommend that you use either Application Load Balancers or Network Load Balancers. However, you can still use a Classic Load Balancer if it supports the features that your application needs. For more information, see Elastic Load Balancing Types (p. 77).

The procedure for deploying this scalable, load-balanced architecture for dynamic traffic consists of the following steps.

- First, choose either the launch template or the launch configuration instructions, based on your preference. When you create a launch template or launch configuration, you include information such as the ID of the Amazon Machine Image (AMI) to use, the instance type, key pair, and block device mapping. The required inputs are the instance type and AMI.
- Next, configure your Auto Scaling group and attach the load balancer.
- Finally, create your Auto Scaling group and verify that your load balancer is attached.

Contents

Prerequisites

- You have an existing VPC and security group to use. If you choose to use the default VPC, after you launch the load balancer you must ensure that its security group has appropriate inbound rules. The default rule does not accept any inbound traffic.
- You have a load balancer created to use. Make sure that you choose the same Availability Zones for the load balancer that you plan to enable for your Auto Scaling group.
- (Optional) You have an IAM role that grants your application the access to AWS that it needs.

- (Optional) You have an Amazon Machine Image (AMI) defined to be the source template for your Amazon EC2 instances. To create one now, launch an instance. Specify the IAM role (if you created one) and any configuration scripts that you need as user data. Connect to the instance and customize it. For example, you can install software and applications on it. Test your applications on your instance to ensure that your instance is configured correctly. Save this updated configuration as a custom AMI. You can terminate the instance if you won't need it later. Instances launched from this new custom AMI include the customizations that you made when you created the AMI.

Deploy Your Application (Console)

The following sections step you through the process of deploying your application.

Tasks

Create or Select a Launch Template

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

To select an existing launch template

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 AWS Region that you used when creating your load balancer.
3. On the navigation pane, choose **Launch Templates**.
4. Select a launch template.
5. Choose **Actions**, **Create Auto Scaling group**.

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

To create a launch template

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 AWS Region that you used when creating your load balancer.
3. On the navigation pane, choose **Launch Templates**.
4. Choose **Create launch template**.
5. Provide a name and description for the launch template. The name of a launch template must have 3 to 125 characters, and can include the following characters: ()./_.
6. For **AMI ID**, enter the ID of the AMI for your instances.
7. For **Instance type**, select a hardware configuration for your instances that is compatible with the AMI that you specified.
8. (Optional) For **Key pair name**, enter the name of the key pair to use when connecting to your instances.
9. For **Network interfaces**, do the following:

   a. Choose **Add network interface**.

    b.   (Optional) To connect to instances in a nondefault VPC, for **Auto-assign public IP**, choose **Enable**.

    c.   For **Security group ID**, specify a security group for your instances.

    d.   For **Delete on termination**, choose whether the network interface is deleted when the Auto Scaling group scales in and terminates the instance to which the network interface is attached.

10. (Optional) To securely distribute credentials to your instances, for **Advanced details**, **IAM instance profile**, enter the Amazon Resource Name (ARN) of your IAM role.

11. (Optional) To specify user data or a configuration script for your instances, paste it into **Advanced details**, **User data**.

12. Choose **Create launch template**.

13. On the confirmation page, choose **Create Auto Scaling group**.

Create or Select a Launch Configuration

If you already have a launch configuration that you'd like to use, select it by 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 AWS Region that you used when creating your load balancer.
3. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
4. On the next page, choose **Create Auto Scaling group**.
5. On the **Create Auto Scaling Group** page, choose **Launch Configuration**, select an existing launch configuration, and then choose **Next Step**.

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 AWS Region that you used when creating your load balancer.
3. On the navigation pane, under **Auto Scaling**, choose **Auto Scaling Groups**.
4. On the next page, choose **Create Auto Scaling group**.
5. On the **Create Auto Scaling Group** page, choose **Launch Configuration**, **Create a new launch configuration**, and then choose **Next Step**.
6. On the **Choose AMI** page, select your custom AMI.
7. On the **Choose Instance Type** page, select a hardware configuration for your instance, and then choose **Next: Configure details**.
8. On the **Configure Details** page, do the following:

    a.   For **Name**, enter 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 instances 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**.

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. Select the acknowledgment check box, and then choose **Create launch configuration**.

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

After completing the instructions above, you're ready to proceed with the wizard to create an Auto Scaling group.

Create an Auto Scaling Group

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

> **Note**
> Amazon EC2 Auto Scaling has changed the Auto Scaling group interface. By default, you're shown the old user interface, but you can switch to the new user interface. This section contains steps for both.
> To switch to the new user interface, on the **Auto Scaling Groups** page, on the banner at the top of the page, choose **Go to the new console**.

To create an Auto Scaling group (new console)

1. From the new console, on the **Auto Scaling groups** page, choose **Create an Auto Scaling group**.

2. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.

3. For **Launch template**, choose one of the following options:

    a. [Launch template only] Choose the launch template that you created and then choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

    b. [Launch configuration only] Choose **Switch to launch configuration** and choose the launch configuration that you created.

4. Choose **Next**.

    The **Configure settings** page appears, allowing you to configure network settings and giving you options for enabling diversification across On-Demand Instances and Spot Instances of multiple instance types (if you chose a launch template).

5. [Launch template only] Keep **Purchase options and instance types** set to **Adhere to the launch template** to use the EC2 instance type and purchase option that are specified in the launch template.

6. For **Network**, choose the VPC that you used for your load balancer and security group.

7. For **Subnet**, choose one or more subnets from each Availability Zone that you want to include.

8. Choose **Next**.

9. On the **Specify load balancing and health checks** page, under **Load balancing**, choose **Enable load balancing**.

10. Do one of the following:

    a. Choose **Application Load Balancer or Network Load Balancer**, and then choose your load balancer target group.

    b. Choose **Classic Load Balancers**, and then choose your load balancer.

11. (Optional) To use Elastic Load Balancing health checks, for **Health checks**, choose **ELB** under **Health check type**.

12. When you have finished configuring the Auto Scaling group, choose **Skip to review**.

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

To create an Auto Scaling group (old console)

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

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

   b. [Launch template] For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

   c. [Launch template] For **Fleet Composition**, choose **Adhere to the launch template** to use the EC2 instance type and purchase option that are specified in the launch template.

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

   e. If you selected an instance type for your launch configuration or template 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.

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

   g. 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**.

      - [Application/Network Load Balancers] Select your target group from **Target Groups**.

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

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

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

3. When you have finished configuring the Auto Scaling group, choose **Review**.

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

After you have created the Auto Scaling group with the load balancer attached, the load balancer automatically registers new instances as they come online. You have only one instance at this point, so there isn't much to register. However, you can now add additional instances by updating the desired capacity of the group. For step-by-step instructions, see Manual Scaling (p. 96).

(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 (new console)

1. Select the check box next to your Auto Scaling group.

   If you're seeing the old user interface, switch to the new user interface by choosing **Go to the new console** from the banner that's displayed at the top of the console page.

2. On the **Details** tab, **Load balancing** shows any attached load balancer target groups or Classic Load Balancers.

3. On the **Activity** tab, under **Activity history**, the **Status** column shows you whether your Auto Scaling group has successfully launched instances.

4. On the **Instance management** tab, under **Instances**, you can view the status of the instances that are currently running.

   The **Lifecycle** column shows the state of your 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.

To verify that your load balancer is attached to your Auto Scaling group (old console)

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 **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.

4. 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.

Deploy Your Application (AWS CLI)

The following sections step you through the process of deploying your application.

Tasks

Create a Launch Template

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

To create the launch template

Use the following **create-launch-template** command and pass a JSON file that contains the information for creating the launch template. To assign public IP addresses to instances in a nondefault VPC, add the network interface attribute and set it to `"NetworkInterfaces": {"AssociatePublicIpAddress":true}` as shown.

```
aws ec2 create-launch-template --launch-template-name my-launch-template --version-
description my-version-description \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-903004f8"],"DeleteOnTermination":true}],"ImageId":"ami-01e24be29428c15b2","InstanceType":"t2.micro
```

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. To assign public IP addresses to instances in a nondefault VPC, use the `--associate-public-ip-address` option as shown.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-launch-config \
  --image-id ami-01e24be29428c15b2 --instance-type t2.micro --associate-public-ip-address \
  --security-groups sg-eb2af88e
```

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. You can use either a launch configuration or a launch template to automatically configure the instances that your Auto Scaling group launches.

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-asg \
  --launch-configuration-name my-launch-config \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --load-balancer-names "my-load-balancer" \
  --max-size 5 --min-size 1 --desired-capacity 2
```

To create an Auto Scaling group with an attached target group for an Application Load Balancer or Network Load Balancer

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-asg \
  --launch-template "LaunchTemplateName=my-launch-template,Version=1" \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --target-group-arns "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-
targets/1234567890123456" \
  --max-size 5 --min-size 1 --desired-capacity 2
```

Next Steps

Now that you have completed this tutorial, you can learn more:

- You can configure your Auto Scaling group to use a scaling policy. This allows the group to scale your Amazon EC2 capacity automatically to handle changes in the amount of traffic that your application receives. For more information, see Dynamic Scaling (p. 108).

- If you would like to learn how to create automated schedules for scaling, Scheduled Scaling (p. 105) provides an introduction to scheduling an Auto Scaling group to scale using one-time or recurring scaling actions.

- You can configure your Auto Scaling group to use Elastic Load Balancing health checks. If you enable load balancer health checks and an instance fails the health checks, the Auto Scaling group considers the instance unhealthy and replaces it. For more information, see Adding ELB Health Checks (p. 80).

- You can expand your application to an additional Availability Zone in the same AWS Region to increase fault tolerance in the event of a service disruption. For more information, see Adding an Availability Zone (p. 81).

Clean Up Your AWS Resources

You've now successfully completed the tutorial. To avoid unnecessary charges to your account for resources that you aren't using, you should clean up the resources that you created just for this tutorial. For step-by-step instructions, see Deleting Your Auto Scaling Infrastructure (p. 90).

Launch Templates

Important
We recommend that you create Auto Scaling groups from launch templates to ensure that you're getting the latest features from Amazon EC2.

A launch template is similar to a launch configuration (p. 36), in that it specifies instance configuration information. Included are the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances. However, defining a launch template instead of a launch configuration allows you to have multiple versions of a template. With versioning, you can create a subset of the full set of parameters and then reuse it to create other templates or template versions. For example, you can create a default template that defines common configuration parameters and allow the other parameters to be specified as part of another version of the same template.

If you plan to continue to use launch configurations with Amazon EC2 Auto Scaling, be aware that not all Auto Scaling group features are available. For example, you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48).

In addition to the features of Amazon EC2 Auto Scaling that you can configure by using launch templates, launch templates provide more advanced Amazon EC2 configuration options. For example, you must use launch templates to use Amazon EC2 Dedicated Hosts. Dedicated Hosts are physical servers with EC2 instance capacity that are dedicated to your use. While Amazon EC2 Dedicated Instances also run on dedicated hardware, the advantage of using Dedicated Hosts over Dedicated Instances is that you can bring eligible software licenses from external vendors and use them on EC2 instances.

If you currently use launch configurations, you can specify a launch template when you update an Auto Scaling group that was created using a launch configuration.

To create a launch template to use with an Auto Scaling group, create the template from scratch, create a new version of an existing template, or copy the parameters from a launch configuration, running instance, or other template.

The following topics describe the most common procedures for creating and working with launch templates for use with your Auto Scaling groups. For more information about launch templates, see the Launching an instance from a launch template section of the *Amazon EC2 User Guide for Linux Instances*.

Contents

Creating a Launch Template for an Auto Scaling Group

Before you can create an Auto Scaling group using a launch template, you must create a launch template that includes the parameters required to launch an EC2 instance, such as the ID of the Amazon Machine Image (AMI) and an instance type.

The following procedure works for creating a new launch template. The new template uses parameters that you define (starting from scratch) or an existing launch template. After you create your launch template, you can create the Auto Scaling group by following the instructions in Creating an Auto Scaling Group Using a Launch Template (p. 64).

Prerequisites

For information about the required IAM permissions, see Controlling the Use of Launch Templates in the *Amazon EC2 User Guide for Linux Instances* and the Launch Template Support (p. 201) topic in this guide.

Considerations

Keep the following considerations in mind when creating a launch template for use with an Auto Scaling group:

- Launch templates give you the flexibility of launching one type of instance, or a combination of instance types and On-Demand and Spot purchase options. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48). Launching instances with such a combination is not supported:
  - If you specify a Spot Instance request in **Additional Details**
  - In EC2-Classic
- If you configure a network type (VPC or EC2-Classic), subnet, and Availability Zone for your template, these settings are ignored in favor of what is specified in the Auto Scaling group.
- If you specify a network interface, you must configure the security group as part of the network interface, and not in the **Security Groups** section of the template.
- You cannot specify multiple network interfaces.
- You cannot assign specific private IP addresses. When an instance launches, a private address is allocated from the CIDR range of the subnet in which the instance is launched. For more information on specifying CIDR ranges for your VPC or subnet, see the Amazon VPC User Guide.
- To specify an existing network interface to use, its device index must be 0 (eth0). For this scenario, you must use the CLI or API to create the Auto Scaling group. When you create the group using the CLI create-auto-scaling-group command or API CreateAutoScalingGroup action, you must specify the Availability Zones parameter instead of the subnet (VPC zone identifier) parameter.
- You cannot use host placement affinity or target a specific host by choosing a host ID.
- Support for host tenancy (Dedicated Hosts) is only available if you specify a host resource group. For more information, see Host Resource Groups in the *AWS License Manager User Guide*. Note that each AMI based on a license configuration association can be mapped to only one host resource group at a time.

To create a new launch template for an Auto Scaling group (new EC2 console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Launch Templates**.
3. Choose **Create launch template**. Enter a name and provide a description for the initial version of the launch template.
4. Under **Launch template contents**, fill out each required field and any optional fields to use as your instance launch specification.

   a. (Required) **Amazon machine image (AMI)**: Choose the ID of the AMI from which to launch the instances. You can search through all available AMIs, or from the **Quick Start** list, select from one of the commonly used AMIs in the list. If you don't see the AMI that you need, you can find a suitable AMI, make note of its ID, and specify it as a custom value.

   b. (Recommended) **Instance type**: Choose the instance type. Ensure that the instance type is compatible with the AMI you've specified.

   c. (Recommended) **Key pair (login)**: Specify the key pair to use when connecting to your instances.
5. Under **Network settings**, to configure one or more VPC or EC2-Classic security groups, complete the following steps:

    a. **Networking platform**: Choose whether to launch instances into a VPC or EC2-Classic, if applicable. However, the network type and Availability Zone settings of the launch template are ignored for Amazon EC2 Auto Scaling in favor of the settings of the Auto Scaling group.

    b. **Security groups**: Choose one or more security groups, or leave blank to configure one or more VPC security groups as part of the network interface. You cannot specify security groups in both places. If you're using EC2-Classic, you must use security groups created specifically for EC2-Classic.

6. Each instance that is launched has an associated root device volume, either an Amazon EBS volume or an instance store volume. You can specify additional EBS volumes or instance store volumes to attach to an instance when it is launched. If you choose to specify volumes to attach to the instances not including the volumes specified by the AMI, under **Storage (Volumes)**, choose **Add new volume** and provide the following information:

    a. **Volume type**: The type of volume depends on the instance type that you've chosen. For more information, see Amazon EC2 Instance Store and Amazon EBS Volumes in the *Amazon EC2 User Guide for Linux Instances*.

    b. **Device name**: Specify a device name for the volume.

    c. **Snapshot**: Enter the ID of the snapshot from which to create the volume.

    d. **Size (GiB)**: For Amazon EBS-backed volumes, specify a storage size. If you're creating the volume from a snapshot and don't specify a volume size, the default is the snapshot size.

    e. **Volume type**: For Amazon EBS volumes, choose the volume type.

    f. **IOPS**: With a Provisioned IOPS SSD volume, enter the maximum number of input/output operations per second (IOPS) that the volume should support.

    g. **Delete on termination**: For Amazon EBS volumes, choose whether to delete the volume when the associated instance is terminated.

    h. **Encrypted**: Choose **Yes** to change the encryption state of an Amazon EBS volume. The default effect of setting this parameter varies with the choice of volume source, as described in the table below. You must in all cases have permission to use the specified CMK. For more information about specifying encrypted volumes, see Amazon EBS Encryption in the *Amazon EC2 User Guide for Linux Instances*.

Encryption Outcomes

If `Encrypted` parameter is set to...	And if source of volume is...	Then the default encryption state is...	Notes
No	New (empty) volume	Unencrypted*	N/A
	Unencrypted snapshot that you own	Unencrypted*	
	Encrypted snapshot that you own	Encrypted by same key	
	Unencrypted snapshot that is shared with you	Unencrypted*	
	Encrypted snapshot that is shared with you	Encrypted by default CMK	
Yes	New volume	Encrypted by default CMK	To use a non-default CMK, specify a value

If `Encrypted` parameter is set to...	And if source of volume is...	Then the default encryption state is...	Notes
	Unencrypted snapshot that you own	Encrypted by default CMK	for the **Key** parameter.
	Encrypted snapshot that you own	Encrypted by same key	
	Unencrypted snapshot that is shared with you	Encrypted by default CMK	
	Encrypted snapshot that is shared with you	Encrypted by default CMK	

* If encryption by default is enabled, all newly created volumes (whether or not the **Encrypted** parameter is set to **Yes**) are encrypted using the default CMK. Setting both the **Encrypted** and **Key** parameters allows you to specify a non-default CMK.

i. (Optional) **Key**: If you chose **Yes** in the previous step, enter the customer master key (CMK) you want to use when encrypting the volumes. Enter any CMK that you previously created using the AWS Key Management Service. You can paste the full ARN of any key that you have access to. For more information, see the AWS Key Management Service Developer Guide and the Required CMK Key Policy for Use with Encrypted Volumes (p. 207) topic in this guide. Note: Amazon EBS does not support asymmetric CMKs.

> **Note**
> Providing a CMK without also setting the **Encrypted** parameter results in an error.

7. For **Instance tags**, specify tags by providing key and value combinations. You can tag the instances, the volumes, or both.

8. Under **Network interfaces**, choose **Add network interface** and provide the following optional information. When launching instances in a VPC, you can specify the default network interface, called the *primary network interface* (eth0). You can only specify one network interface. Pay attention to the following fields:

    a. **Device**: Specify `eth0` as the device name (the device for which the device index is 0).

    b. **Network interface**: Leave blank to let AWS create a new network interface when an instance is launched, or enter the ID of an existing network interface. If you specify an ID, this limits your Auto Scaling group to one instance.

    c. **Description**: Enter a descriptive name.

    d. **Subnet**: While you may choose to specify a subnet, it is ignored for Amazon EC2 Auto Scaling in favor of the settings of the Auto Scaling group.

    e. **Auto-assign public IP**: Specify whether to automatically assign a public IP address to the network interface with the device index of `eth0`. This setting can only be enabled for a single, new network interface. For more information, see Assigning a Public IPv4 Address During Instance Launch in the *Amazon EC2 User Guide for Linux Instances*.

    f. **Security group ID**: Enter the IDs of one or more security groups with which to associate the primary network interface (eth0). Each security group must be configured for the VPC that your Auto Scaling group will launch instances into. Separate the entries with commas.

    g. **Delete on termination**: Choose whether the network interface is deleted when the Auto Scaling group scales in and terminates the instance to which the network interface is attached.

9. For **Advanced Details**, expand the section to view the fields and specify any additional parameters for the instances.

- **Purchasing option**: You have the option to request Spot Instances and specify the maximum price you are willing to pay per instance hour. For this to work with an Auto Scaling group, you must specify a one-time request with no end date. For more information, see Launching Spot Instances in Your Auto Scaling Group (p. 84).

  > **Important**
  > If you plan to specify multiple instance types and purchase options when you configure your Auto Scaling group, leave these fields empty. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48).

- **IAM instance profile**: Specify an AWS Identity and Access Management (IAM) instance profile to associate with the instances. For more information, see IAM Role for Applications That Run on Amazon EC2 Instances (p. 206).

- **Shutdown behavior**: You can leave this field blank because it is ignored for Amazon EC2 Auto Scaling. The default behavior of Amazon EC2 Auto Scaling is to terminate the instance.

- **Stop - Hibernate behavior**: You can leave this field blank because it is ignored for Amazon EC2 Auto Scaling. The default behavior of Amazon EC2 Auto Scaling is to terminate the instance.

- **Termination protection**: Provides additional termination protection but is ignored for Amazon EC2 Auto Scaling when scaling in your Auto Scaling group. To control whether an Auto Scaling group can terminate a particular instance when scaling in, use Instance Scale-In Protection (p. 140).

- **Detailed CloudWatch monitoring**: Choose whether to enable detailed monitoring of the instances using Amazon CloudWatch. Additional charges apply. For more information, see Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch (p. 173).

- **T2/T3 Unlimited**: (Only valid for T2 and T3 instances) Choose whether to enable applications to burst beyond the baseline for as long as needed. Additional charges may apply. For more information, see Using an Auto Scaling Group to Launch a Burstable Performance Instance as Unlimited in the *Amazon EC2 User Guide for Linux Instances*.

- **Placement group name**: Specify a placement group in which to launch the instances. Not all instance types can be launched in a placement group. If you configure an Auto Scaling group using a CLI command that specifies a different placement group, the setting is ignored in favor of the one specified for the Auto Scaling group.

- **EBS-optimized instance**: Provides additional, dedicated capacity for Amazon EBS I/O. Not all instance types support this feature, and additional charges apply.

- **Tenancy**: You can choose to run your instances on shared hardware (**Shared**), on dedicated hardware (**Dedicated**), or, when using a host resource group, on Dedicated Hosts (**Dedicated host**). Additional charges may apply.

- **Tenancy host resource group**: Specify a host resource group for a BYOL AMI to use on Dedicated Hosts. For more information, see Host Resource Groups in the *AWS License Manager User Guide*. Note: You do not have to have already allocated Dedicated Hosts in your account before you use this feature. Your instances will automatically launch onto Dedicated Hosts regardless.

- **RAM disk ID**: The ID of the RAM disk associated with the AMI. Only valid for paravirtual (PV) AMIs.

- **Kernel ID**: The ID of the kernel associated with the AMI. Only valid for paravirtual (PV) AMIs.

- **License configurations**: You can specify the license configuration to use.

- **User data**: You can specify user data to configure an instance during launch, or to run a configuration script.

10. Choose **Create launch template**.

If you are not currently using the new EC2 console, you can create a launch template using the old EC2 console instead.

To create a new launch template for an Auto Scaling group (old EC2 console)

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

2. In the navigation pane, choose **Launch Templates**.

3. Choose **Create a new template**. Enter a name and provide a description for the initial version of the launch template.

4. If you choose to create the new template based on another template:

   a. For **Source template**, choose an existing launch template.

   b. Choose the launch template version on which to base the new launch template from **Source template version**.

5. Under **Launch template contents**, provide the following information.

   a. **AMI ID**: Choose the ID of the Amazon Machine Image (AMI) from which to launch the instances. You can search through all available AMIs using the **Search for AMI** dialog. From the **Quick Start** list, select from one of the commonly used AMIs in the list. If you don't see the AMI that you need, select the **AWS Marketplace** or **Community AMIs** list to find a suitable AMI.

   b. **Instance type**: Choose the instance type. Ensure that the instance type is compatible with the AMI you've specified.

   c. **Key pair name**: Specify the key pair to use when connecting to your instances.

   d. **Network type**: You can choose to specify whether to launch instances into a VPC or EC2-Classic, if applicable. However, the network type and Availability Zone settings of the launch template are ignored for Amazon EC2 Auto Scaling in favor of the settings of the Auto Scaling group.

   e. **Security Groups**: Choose one or more VPC or EC2-Classic security groups, or leave blank to configure one or more VPC security groups as part of the network interface. You cannot specify security groups in both places. If you're using EC2-Classic, you must use security groups created specifically for EC2-Classic.

6. Under **Network interfaces**, specify the primary network interface (eth0). You can only specify one network interface.

7. Under **Storage (Volumes)**, specify additional storage volumes for your instance, using a block device mapping.

8. For **Instance tags**, specify tags by providing key and value combinations. You can tag the instances, the volumes, or both.

9. For **Advanced Details**, expand the section to view the fields and specify any additional parameters for the instances.

10. Choose **Create launch template**.

To create a launch template from an existing instance

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

2. In the navigation pane, choose **Instances**.

3. Select the instance and choose **Actions**, **Create Template from Instance**.

4. Provide a name and description. Adjust any other launch parameters as required, and choose **Create Launch Template**.

To create a launch template using the command line

You can use one of the following commands:

- **create-launch-template** (AWS CLI)
- **New-EC2LaunchTemplate** (AWS Tools for Windows PowerShell)

Create a launch template using the **create-launch-template** command as follows. Specify a value for `Groups` that corresponds to security groups for the VPC that your Auto Scaling group will launch instances into. Specify the VPC and subnets as properties of the Auto Scaling group.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-7c227019"],"DeleteOnTermination":true}],"ImageId":"ami-01e24be29428c15b2","InstanceType":"t2.micro
 [{"ResourceType":"instance","Tags":[{"Key":"purpose","Value":"webserver"}]}]}'
```

Use the following **describe-launch-templates** command to describe the launch template *my-template-for-auto-scaling*.

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

Use the following **describe-launch-template-versions** command to describe the versions of the specified launch template *my-template-for-auto-scaling*.

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b72934aff71
```

The following is an example response:

```
{
    "LaunchTemplateVersions": [
        {
            "VersionDescription": "version1",
            "LaunchTemplateId": "lt-068f72b72934aff71",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "LaunchTemplateData": {
                "TagSpecifications": [
                    {
                        "ResourceType": "instance",
                        "Tags": [
                            {
                                "Key": "purpose",
                                "Value": "webserver"
                            }
                        ]
                    }
                ],
                "ImageId": "ami-01e24be29428c15b2",
                "InstanceType": "t2.micro",
                "NetworkInterfaces": [
                    {
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "Groups": [
                            "sg-7c227019"
                        ],
                        "AssociatePublicIpAddress": true
                    }
                ]
            },
            "DefaultVersion": true,
            "CreateTime": "2019-02-28T19:52:27.000Z"
        }
    ]
```

```
}
```

Copying a Launch Configuration to a Launch Template

Use the following procedure to copy the options from an existing launch configuration to create a new launch template. This action can only be performed from the console.

You can create launch templates from existing launch configurations to make it easy for you to update your Auto Scaling groups to use launch templates. Like launch configurations, launch templates can contain all or some of the parameters to launch an instance. With launch templates, you can also create multiple versions of a template to make it faster and easier to launch new instances.

To create a launch template from a launch configuration

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Launch Configurations**.
3. Select the launch configuration you want to copy and choose **Copy to launch template**. This sets up a new launch template with the same name and options as the launch configuration you selected.
4. For **New launch template name**, you can use the name of the launch configuration (the default) or enter a new name. Launch template names must be unique.
5. (Optional) To create an Auto Scaling group using the new launch template, select **Create an Auto Scaling group using the new template**. For more information, see Creating an Auto Scaling Group Using a Launch Template (p. 64).
6. Choose **Submit**.

After creating your launch template, you can update your existing Auto Scaling groups, as needed, with the launch template that you created. For more information, see Replacing a Launch Configuration with a Launch Template (p. 34).

Replacing a Launch Configuration with a Launch Template

When you edit an Auto Scaling group that has an existing launch configuration, you have the option of replacing the launch configuration with a launch template. This lets you use launch templates with any Auto Scaling groups that you currently use. In doing so, you can take advantage of the versioning and other features of launch templates.

After you replace the launch configuration for an Auto Scaling group, any new instances are launched using the new launch template, but existing instances are not affected. In this situation, you can terminate existing instances in the Auto Scaling group to force a new instance to launch that uses your launch template. Or, you can allow automatic scaling to gradually replace older instances with newer instances based on your termination policies (p. 137). You can also automate deployment of the update with a few clicks through AWS CloudFormation.

Prerequisites

Before you can replace a launch configuration in an Auto Scaling group, you must first create your launch template. The easiest way to create a launch template is to copy it from the launch configuration. For more information, see Copying a Launch Configuration to a Launch Template (p. 34).

When you replace a launch configuration with a launch template, your `ec2:RunInstances` permissions are checked. If you are attempting to use a launch template and you do not have sufficient permissions, you receive an error that you're not authorized to use the launch template.

To replace the launch configuration for an Auto Scaling group

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Auto Scaling Groups**.
3. Select the Auto Scaling group and choose **Details, Edit**.
4. For **Launch Instances Using**, choose the **Launch Template** option.
5. For **Launch Template**, select your launch template.
6. For **Launch Template Version**, select the launch template version, as needed. After you create versions of a launch template, you can choose whether the Auto Scaling group uses the default or the latest version of the launch template when scaling out.
7. When you have finished, choose **Save**.

To replace a launch configuration using the command line

You can use one of the following commands:

- **update-auto-scaling-group** (AWS CLI)
- **Update-ASAutoScalingGroup** (AWS Tools for Windows PowerShell)

Launch Configurations

A *launch configuration* is an instance configuration template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances. Include 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.

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. To change the launch configuration for an Auto Scaling group, you must create a launch configuration and then update your Auto Scaling group with it.

Keep in mind that whenever you create an Auto Scaling group, you must specify a launch configuration, a launch template, or an EC2 instance. When you create an Auto Scaling group using an EC2 instance, Amazon EC2 Auto Scaling automatically creates a launch configuration for you and associates it with the Auto Scaling group. For more information, see Creating an Auto Scaling Group Using an EC2 Instance (p. 69). Alternatively, if you are using launch templates, you can specify a launch template instead of a launch configuration or an EC2 instance. For more information, see Launch Templates (p. 27).

Contents

Creating a Launch Configuration

When you create a launch configuration, you must specify information about the EC2 instances to launch. Include the Amazon Machine Image (AMI), instance type, key pair, security groups, and block device mapping. Alternatively, you can create a launch configuration using attributes from a running EC2 instance. For more information, see Creating a Launch Configuration Using an EC2 Instance (p. 37).

> **Note**
> When you create an Auto Scaling group, you can specify a launch template, launch configuration, or an EC2 instance. We recommend that you use a launch template to ensure that you can use the latest features of Amazon EC2. For more information, see Launch Templates (p. 27).

After you create a launch configuration, you can create an Auto Scaling group. For more information, see Creating an Auto Scaling Group Using a Launch Configuration (p. 67).

An Auto Scaling group is associated with one launch configuration 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 an existing Auto Scaling group, you must update it with the new launch configuration. For more information, see Changing the Launch Configuration for an Auto Scaling Group (p. 41).

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**.

4.   On the next page, 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.   (Optional) For **Purchasing option**, you may request Spot Instances and specify the maximum price you are willing to pay per instance hour. For more information, see Launching Spot Instances in Your Auto Scaling Group (p. 84).

   c.   (Optional) For **IAM role**, select a role to associate with the instances. For more information, see IAM Role for Applications That Run on Amazon EC2 Instances (p. 206).

   d.   (Optional) By default, basic monitoring is enabled for your Auto Scaling instances. To enable detailed monitoring for your Auto Scaling instances, select **Enable CloudWatch detailed monitoring**.

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

   f.   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**
> If you need to connect to your instance, do not select **Proceed without a key pair**.

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

Amazon EC2 Auto Scaling provides you with an option to create a launch configuration using the attributes from a running EC2 instance.

> **Tip**
> You can create an Auto Scaling group directly from an EC2 instance (p. 69). When you use this feature, Amazon EC2 Auto Scaling automatically creates a launch configuration for you as well.

If the specified instance has properties that are not currently supported by launch configurations, the instances launched by the Auto Scaling group might not be identical to the original EC2 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, Amazon EC2 Auto Scaling derives attributes for the launch configuration from the specified instance. Attributes are also derived from the block device mapping for the AMI from which the instance was launched, ignoring any additional block devices that were added after launch.

When you create a launch configuration using a running instance, you can override the following attributes by specifying them 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 (p. 38)
- Create a Launch Configuration from an Instance and Override the Block Devices (AWS CLI) (p. 39)
- Create a Launch Configuration and Override the Instance Type (AWS CLI) (p. 40)

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 (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. 98).

Create a Launch Configuration from an EC2 Instance (AWS CLI)

Use the following **create-launch-configuration** command to create a launch configuration from an instance using the same attributes as the instance. Any block devices added after launch 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,
            "LaunchConfigurationARN": "arn",
```

Amazon EC2 Auto Scaling User Guide
Create a Launch Configuration from an Instance
and Override the Block Devices (AWS CLI)

```
        "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 (AWS CLI)

By default, Amazon EC2 Auto Scaling uses the attributes from the EC2 instance that you specify to create the launch configuration. However, 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

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 (AWS CLI)

By default, Amazon EC2 Auto Scaling uses the attributes from the EC2 instance you specify to create the launch configuration. Depending on your requirements, you might want 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

Use the following **create-launch-configuration** command to create a launch configuration using an EC2 instance but with a different instance type (for example `t2.medium`) than the instance (for example `t2.micro`):

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-
instance-changetype \
  --instance-id i-a8e09d9c --instance-type t2.medium
```

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": "t2.medium",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

Changing the Launch Configuration for an Auto Scaling Group

An Auto Scaling group is associated with one launch configuration at a time, and you can't modify a launch configuration after you've created it. To change the launch configuration for an Auto Scaling group, use an existing launch configuration as the basis for a new launch configuration. Then, update the Auto Scaling group to use the new launch configuration.

After you change the launch configuration for an Auto Scaling group, any new instances are launched using the new configuration options, but existing instances are not affected. In this situation, you can terminate existing instances in the Auto Scaling group to force a new instance to launch that uses the new configuration. Or, you can allow automatic scaling to gradually replace older instances with newer instances based on your termination policies (p. 137). You can also automate deployment of the updated launch configuration with a few clicks through AWS CloudFormation.

To change the launch configuration for an Auto Scaling group (console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Launch Configurations**.
3. Select the launch configuration and choose **Actions**, **Copy launch configuration**. This sets up a new launch configuration with the same options as the original, but with "Copy" added to the name.
4. On the **Copy Launch Configuration** page, edit the configuration options as needed and choose **Create launch configuration**.
5. On the confirmation page, choose **View your Auto Scaling groups**.
6. Select the Auto Scaling group and choose **Details**, **Edit**.
7. Select the new launch configuration from **Launch Configuration** and choose **Save**.

To change the launch configuration for an Auto Scaling group (AWS CLI)

1. Describe the current launch configuration using the **describe-launch-configurations** command.

2. Create a new launch configuration using the **create-launch-configuration** command.

3. Update the launch configuration for the Auto Scaling group using the **update-auto-scaling-group** command with the `--launch-configuration-names` parameter.

To change the launch configuration for an Auto Scaling group (Tools for Windows PowerShell)

1. Describe the current launch configuration using the **Get-ASLaunchConfiguration** command.

2. Create a new launch configuration using the **New-ASLaunchConfiguration** command.

3. Update the launch configuration for the Auto Scaling group using the **Update-ASAutoScalingGroup** command with the `-LaunchConfigurationName` parameter.

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.

Before you can launch your Auto Scaling instances in a new 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 a private IP address in the address range of the subnet. This enables your instances to communicate with other instances in the VPC.

You can configure your launch configuration to assign public IP addresses to your instances. Assigning public IP addresses to your instances enables them to communicate with the internet or other services in AWS.

When you enable public IP addresses for your instances and launch them into a subnet that is configured to automatically assign IPv6 addresses, they receive both IPv4 and IPv6 addresses. Otherwise, they receive only IPv4 addresses. For more information, see IPv6 Addresses in the *Amazon EC2 User Guide for Linux Instances*.

Instance Placement Tenancy

Tenancy defines how EC2 instances are distributed across physical hardware and affects pricing. There are three tenancy options available:

- Shared (`default`) – Multiple AWS accounts may share the same physical hardware.
- Dedicated instance (`dedicated`) – Your instance runs on single-tenant hardware.
- Dedicated host (`host`) – Your instance runs on a dedicated host, which is an isolated server with configurations that you can control.

> **Important**
> You can configure tenancy for EC2 instances using a launch configuration or launch template. However, the `host` tenancy value cannot be used with a launch configuration. Use the `default` or `dedicated` tenancy values only. To use a tenancy value of `host`, you must use a launch template. For more information, see Creating a Launch Template for an Auto Scaling Group (p. 27).

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 EC2 User Guide for Linux Instances*. 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

To configure tenancy using a launch configuration (AWS CLI)

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. The `PlacementTenancy` parameter is only part of the output for this command when you 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. It enables 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 the Auto Scaling group launch new instances that are linked to the VPC.

Link to a VPC (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

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. Add 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**.
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 (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

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. Add 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. Specify a value for *vpd_id* as the ID of the VPC with ClassicLink enabled from step 1 and for *group_id* as 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**.

Auto Scaling Groups

An *Auto Scaling group* contains a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. An Auto Scaling group also enables you to use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. Both maintaining the number of instances in an Auto Scaling group and automatic scaling are the core functionality of the Amazon EC2 Auto Scaling service.

The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity. You can adjust its size to meet demand, either manually or by using automatic scaling.

An Auto Scaling group starts by launching enough instances to meet its desired capacity. It maintains this number of instances by performing periodic health checks on the instances in the group. The Auto Scaling group continues to maintain a fixed number of instances even if an instance becomes unhealthy. If an instance becomes unhealthy, the group terminates the unhealthy instance and launches another instance to replace it. For more information, see Health Checks for Auto Scaling Instances (p. 160).

You can use scaling policies to increase or decrease the number of instances in your group dynamically to meet changing conditions. When the scaling policy is in effect, the Auto Scaling group adjusts the desired capacity of the group, between the minimum and maximum capacity values that you specify, and launches or terminates the instances as needed. You can also scale on a schedule. For more information, see Scaling the Size of Your Auto Scaling Group (p. 94).

An Auto Scaling group can launch On-Demand Instances, Spot Instances, or both. You can specify multiple purchase options for your Auto Scaling group only when you configure the group to use a launch template. (We recommend that you use launch templates instead of launch configurations to make sure that you can use the latest features of Amazon EC2.)

Spot Instances provide you with access to unused Amazon EC2 capacity at steep discounts relative to On-Demand prices. For more information, see Amazon EC2 Spot Instances. There are key differences between Spot Instances and On-Demand Instances:

- The price for Spot Instances varies based on demand
- Amazon EC2 can terminate an individual Spot Instance as the availability of, or price for, Spot Instances changes

When a Spot Instance is terminated, the Auto Scaling group attempts to launch a replacement instance to maintain the desired capacity for the group.

When instances are launched, if you specified multiple Availability Zones, the desired capacity is distributed across these Availability Zones. If a scaling action occurs, Amazon EC2 Auto Scaling automatically maintains balance across all of the Availability Zones that you specify.

This section provides detailed steps for creating an Auto Scaling group. If you're new to Auto Scaling groups, start with Getting Started with Amazon EC2 Auto Scaling (p. 12) to learn about the various building blocks that are used in Amazon EC2 Auto Scaling.

Contents

Auto Scaling Groups with Multiple Instance Types and Purchase Options

You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for Amazon EC2 instances, while making sure that you obtain the desired scale and performance for your application.

You first specify the common configuration parameters in a launch template, and choose it when you create an Auto Scaling group. When you configure the Auto Scaling group, you can:

- Choose one or more instance types for the group (optionally overriding the instance type that is specified by the launch template).
- Assign each instance type an individual weight. Doing so might be useful, for example, if the instance types offer different vCPU, memory, storage, or network bandwidth capabilities.
- Specify how much On-Demand and Spot capacity to launch, and specify an optional On-Demand base portion.
- Prioritize instance types that can benefit from Reserved Instance or Savings Plan discount pricing.
- Define how Amazon EC2 Auto Scaling should distribute your Spot capacity across instance types.

You enhance availability by deploying your application across multiple instance types running in multiple Availability Zones. You can use just one instance type, but it is a best practice to use a few instance types to allow Amazon EC2 Auto Scaling to launch another instance type in the event that there is insufficient instance capacity in your chosen Availability Zones. With Spot Instances, if there is insufficient instance capacity, Amazon EC2 Auto Scaling keeps trying in other Spot Instance pools (determined by your choice of instance types and allocation strategy) rather than launching On-Demand Instances, so that you can leverage the cost savings of Spot Instances.

Allocation Strategies

The following allocation strategies determine how the Auto Scaling group fulfills On-Demand and Spot capacity from the possible instance types.

In each case, Amazon EC2 Auto Scaling first tries to ensure that your instances are evenly balanced across the Availability Zones that you specified. Then, it launches instance types according to the allocation strategy that is specified.

On-Demand Instances

The allocation strategy for On-Demand Instances is `prioritized`. Amazon EC2 Auto Scaling uses the order of instance types in the list of launch template overrides to determine which instance type to use first when fulfilling On-Demand capacity. For example, let's say that you specified three launch template overrides in the following order: `c5.large`, `c4.large`, and `c3.large`. When your On-Demand Instances are launched, the Auto Scaling group fulfills On-Demand capacity by starting with `c5.large`, then `c4.large`, and then `c3.large`.

Consider the following when managing the priority order of your On-Demand Instances:

You can pay for usage upfront to get significant discounts for On-Demand Instances by using either Reserved Instances or Savings Plans. For more information about Reserved Instances or Savings Plans, see the EC2 pricing page.

- With Reserved Instances, your discounted rate of the regular On-Demand Instance pricing applies if Amazon EC2 Auto Scaling launches matching instance types. That means that if you have unused Reserved Instances for `c4.large`, you can set the instance type priority to give the highest priority for your Reserved Instances to a `c4.large` instance type. When a `c4.large` instance launches, you receive the Reserved Instance pricing.
- With Savings Plans, your discounted rate of the regular On-Demand Instance pricing applies when using either Amazon EC2 Instance Savings Plans or Compute Savings Plans. Because of the flexible nature of Savings Plans, you have greater flexibility in prioritizing your instance types. As long as you use instance types that are covered by your Savings Plan, you can set them in any order of priority and even occasionally change their order entirely, and continue to receive the discounted rate provided by your Savings Plan. To learn more about Savings Plans, see the Savings Plans User Guide.

Spot Instances

Amazon EC2 Auto Scaling provides two types of allocation strategies that can be used for Spot Instances:

`capacity-optimized`

Amazon EC2 Auto Scaling allocates your instances from the Spot Instance pool with optimal capacity for the number of instances that are launching. Deploying in this way helps you make the most efficient use of spare EC2 capacity.

With Spot Instances, the pricing changes slowly over time based on long-term trends in supply and demand, but capacity fluctuates in real time. The `capacity-optimized` strategy automatically launches Spot Instances into the most available pools by looking at real-time capacity data and predicting which are the most available. This works well for workloads such as big data and analytics, image and media rendering, and machine learning. It also works well for high performance computing that may have a higher cost of interruption associated with restarting work and checkpointing. By offering the possibility of fewer interruptions, the `capacity-optimized` strategy can lower the overall cost of your workload.

`lowest-price`

Amazon EC2 Auto Scaling allocates your instances from the number (N) of Spot Instance pools that you specify and from the pools with the lowest price per unit at the time of fulfillment.

For example, if you specify 4 instance types and 4 Availability Zones, your Auto Scaling group can potentially draw from as many as 16 different Spot Instance pools. If you specify 2 Spot pools (N=2) for the allocation strategy, your Auto Scaling group can fulfill Spot capacity from a minimum of 8 different Spot pools where the price per unit is the lowest.

To get started, we recommend choosing the `capacity-optimized` allocation strategy and specifying a few instance types that are appropriate for your application. In addition, you can define a range of Availability Zones for Amazon EC2 Auto Scaling to choose from when launching instances.

Optionally, you can specify a maximum price for your Spot Instances. If you don't specify a maximum price, the default maximum price is the On-Demand price, but you still receive the steep discounts provided by Spot Instances. These discounts are possible because of the stable Spot pricing that is made available using the new Spot pricing model.

For more information about the allocation strategies for Spot Instances, see Introducing the capacity-optimized allocation strategy for Amazon EC2 Spot Instances in the AWS blog.

Controlling the Proportion of On-Demand Instances

You have full control over the proportion of instances in the Auto Scaling group that are launched as On-Demand Instances. To ensure that you always have instance capacity, you can designate a percentage of the group to launch as On-Demand Instances and, optionally, a base number of On-Demand Instances to start with. If you choose to specify a base capacity of On-Demand Instances, the Auto Scaling group ensures that this base capacity of On-Demand Instances is launched first when the group scales out. Anything beyond the base capacity uses the On-Demand percentage to determine how many On-Demand Instances and Spot Instances to launch. You can specify any number from 0 to 100 for the On-Demand percentage.

The behavior of the Auto Scaling group as it increases in size is as follows:

Example: Scaling behavior

Instances Distribution	Total Number of Running Instances Across Purchase Options			
	10	20	30	40
Example 1				
On-Demand base: 10	10	10	10	10
On-Demand percentage above base: 50%	0	5	10	15
Spot percentage: 50%	0	5	10	15
Example 2				
On-Demand base: 0	0	0	0	0
On-Demand percentage above base: 0%	0	0	0	0
Spot percentage: 100%	10	20	30	40
Example 3				
On-Demand base: 0	0	0	0	0
On-Demand percentage above base: 60%	6	12	18	24
Spot percentage: 40%	4	8	12	16
Example 4				
On-Demand base: 0	0	0	0	0

Instances Distribution	Total Number of Running Instances Across Purchase Options			
On-Demand percentage above base: 100%	10	20	30	40
Spot percentage: 0%	0	0	0	0
Example 5				
On-Demand base: 12	10	12	12	12
On-Demand percentage above base: 0%	0	0	0	0
Spot percentage: 100%	0	8	18	28

Best Practices for Spot Instances

Before you create your Auto Scaling group to request Spot Instances, review Spot Best Practices. Use these best practices when you plan your request so that you can provision the type of instances that you want at the lowest possible price. We also recommend that you do the following:

- Use the default maximum price, which is the On-Demand price. You pay only the Spot price for the Spot Instances that you launch. If the Spot price is within your maximum price, whether your request is fulfilled depends on availability. For more information, see Pricing and Savings in the *Amazon EC2 User Guide for Linux Instances*.

- Create your Auto Scaling group with multiple instance types. Because prices fluctuate independently for each instance type in an Availability Zone, you can often get more compute capacity for the same price when you have instance type flexibility.

- Similarly, don't limit yourself to only the most popular instance types. Because prices adjust based on long-term demand, popular instance types (such as recently launched instance families), tend to have more price adjustments. Picking older-generation instance types that are less popular tends to result in lower costs and fewer interruptions.

- If you chose the `lowest-price` allocation strategy and you run a web service, specify a high number of Spot pools, for example, N=10. Specifying a high number of Spot pools reduces the impact of Spot Instance interruptions if a pool in one of the Availability Zones becomes temporarily unavailable. If you run batch processing or other non-mission critical applications, you can specify a lower number of Spot pools, for example, N=2. This helps to ensure that you provision Spot Instances from only the very lowest priced Spot pools available per Availability Zone.

- Use Spot Instance interruption notices to monitor the status of your Spot Instances. For example, you can set up a rule in Amazon CloudWatch Events that automatically sends the EC2 Spot two-minute warning to an Amazon SNS topic, an AWS Lambda function, or another target. For more information, see Spot Instance Interruption Notices in the *Amazon EC2 User Guide for Linux Instances* and the Amazon CloudWatch Events User Guide.

If you intend to specify a maximum price, use the AWS CLI or AWS SDKs to create the Auto Scaling group, but be cautious. If your maximum price is lower than the Spot price for the instance types that you selected, your Spot Instances are not launched.

Prerequisites

Your launch template is configured for use with an Auto Scaling group. For more information, see
Creating a Launch Template for an Auto Scaling Group (p. 27).

IAM users can create an Auto Scaling group using a launch template only if they have permissions to call
the `ec2:RunInstances` action. For more information, see Launch Template Support (p. 201).

Creating an Auto Scaling Group (Console)

Follow these steps to create a fleet of On-Demand Instances and Spot Instances that you can scale.

Amazon EC2 Auto Scaling has changed the Auto Scaling group interface. By default, you're shown the
old user interface, but you can switch to the new user interface. This topic contains steps for both.

To create an Auto Scaling group with multiple purchase options (new 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, choose the same AWS Region that you used when
   you created the launch template.
3. In the navigation pane, choose **Auto Scaling Groups**.
4. Choose **Create an Auto Scaling group**.
5. On the **Choose launch template or configuration** page, do the following:

    a. For **Auto Scaling group name**, enter a name for your Auto Scaling group.

    b. For **Launch template**, choose an existing launch template.

    c. For **Launch template version**, choose whether the Auto Scaling group uses the default, the
       latest, or a specific version of the launch template when scaling out.

    d. Verify that your launch template supports all of the options that you are planning to use, and
       then choose **Next**.

6. On the **Configure settings** page, for **Purchase options and instance types**, choose **Combine
   purchase options and instance types**.
7. Under **Instances distribution**, do the following:

    You can skip these steps if you want to keep the default settings.

    a. For **Optional On-Demand base**, specify the minimum number of instances for the Auto Scaling
       group's initial capacity that must be fulfilled by On-Demand Instances.

    b. For **On-Demand percentage above base**, specify the percentages of On-Demand Instances and
       Spot Instances for your additional capacity beyond the optional On-Demand base amount.

    c. For **Spot allocation strategy per Availability Zone**, we recommend that you keep the default
       setting of **Capacity optimized**. If you prefer not to keep the default, choose **Lowest price**, and
       then enter the number of lowest priced Spot Instance pools to diversify across.

8. For **Instance types**, choose which types of instances can be launched, using our recommendations as
   a starting point. Otherwise, you can delete the instance types and add them later as needed.
9. (Optional) To change the order of the instance types, use the arrows. The order in which you set the
   instance types sets their priority for On-Demand Instances. The instance type at the top of the list is
   prioritized the highest when the Auto Scaling group launches your On-Demand capacity.
10. (Optional) To use instance weighting (p. 58), assign each instance type a relative weight that
    corresponds to how much the instance should count toward the capacity of the Auto Scaling group.
11. Under **Network**, for **VPC**, choose the VPC for the security groups that you specified in your launch
    template. Launching instances using multiple instance types and purchase options is not supported
    in EC2-Classic.

12. For **Subnet**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

13. Choose **Next**.

    Or, you can accept the rest of the defaults, and choose **Skip to review**.

14. On the **Specify load balancing and health checks** page, configure the following options, and then choose **Next**:

    a. (Optional) To register your Amazon EC2 instances with an Elastic Load Balancing (ELB) load balancer, choose **Enable load balancing**. To attach an Application Load Balancer or Network Load Balancer, choose an existing target group or create a new one. To attach a Classic Load Balancer, choose an existing load balancer or create a new one.

    b. (Optional) To enable your ELB health checks, for **Health checks**, choose **ELB** under **Health check type**.

    c. (Optional) Under **Health check grace period**, enter the amount of time until Amazon EC2 Auto Scaling checks the health of instances after they are put into service. The intention is to prevent Amazon EC2 Auto Scaling from marking instances as unhealthy and terminating them before they have time to come up. The default is 300 seconds.

15. On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**:

    a. (Optional) For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see Setting Capacity Limits for Your Auto Scaling Group (p. 95).

    b. (Optional) To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see Target Tracking Scaling Policies (p. 113).

    c. (Optional) Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance Scale-In Protection (p. 140).

16. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see Getting Amazon SNS Notifications When Your Auto Scaling Group Scales (p. 162).

17. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

To create an Auto Scaling group with multiple purchase options (old 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, choose the same AWS Region that you used when you created the launch template.

3. In the navigation pane, choose **Auto Scaling Groups**.

4. Choose **Create Auto Scaling group**.

5. Choose **Launch Template**, choose your launch template, and then choose **Next Step**.

6. On the **Configure Auto Scaling group details** page, for **Group name**, enter a name for your Auto Scaling group.

7. For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

8. For **Fleet Composition**, choose **Combine purchase options and instances** to launch instances across multiple instance types using both On-Demand and Spot purchase options.

9. For **Instance Types**, do the following:

   a. Choose the instance types to use as overrides. By default, a single instance type is specified by the chosen launch template, but it can be changed if you want to specify a different instance type. When choosing multiple instance types, the order in which you add instance types sets their priority for On-Demand Instances. The instance type at the top of the list is prioritized the highest when the Auto Scaling group launches your On-Demand capacity.

   b. (Optional) To use instance weighting (p. 58), assign each instance type a relative weight that corresponds with how much the instance should count toward the desired capacity of the Auto Scaling group.

10. For **Instances Distribution**, choose whether to keep or edit the default settings. Clearing the box for **Use the default settings to get started quickly** allows you to edit the default settings.

11. If you chose to edit the default settings, provide the following information.

    - For **Maximum Spot Price**, choose **Use default** to cap your maximum Spot price at the On-Demand price. Or choose **Set your maximum price** and enter a value to specify the maximum price you are willing to pay per instance per hour for Spot Instances.

    - For **Spot Allocation Strategy**, the default is **Launch Spot Instances optimally based on the available Spot capacity per Availability Zone**. To use the lowest price strategy instead, choose **Diversify Spot Instances across your N lowest priced instance types per Availability Zone**, and then enter the number of Spot pools to use, or accept the default (2).

    - For **Optional On-Demand Base**, specify the minimum number of instances for the Auto Scaling group's initial capacity that must be fulfilled by On-Demand Instances.

    - For **On-Demand Percentage Above Base**, specify the percentages of On-Demand Instances and Spot Instances for your additional capacity beyond the optional On-Demand base amount.

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

13. For **Network**, choose a VPC for your Auto Scaling group. Launching instances using multiple instance types and purchase options is not supported in EC2-Classic.

14. For **Subnet**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

15. (Optional) To register your Amazon EC2 instances with a load balancer, choose **Advanced Details**, choose **Receive traffic from one or more load balancers**, and choose one or more Classic Load Balancers or target groups.

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

17. On the **Configure scaling policies** page, choose one of the following options, and then choose **Next: Configure Notifications**:

    - To manually adjust the size of the Auto Scaling group as needed, choose **Keep this group at its initial size**. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

    - To automatically adjust the size of the Auto Scaling group based on criteria that you specify, choose **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 113).

18. (Optional) To receive notifications, choose **Add notification**, configure the notification, and then choose **Next: Configure Tags**.

19. (Optional) To add tags, choose **Edit tags**, provide a tag key and value for each tag, and then choose **Review**.

    Alternatively, you can add tags later on. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

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

Creating an Auto Scaling Group (AWS CLI)

The following examples show how to create an Auto Scaling group with multiple purchase options using the AWS CLI **create-auto-scaling-group** command.

Example: Launch Spot Instances using the `capacity-optimized` allocation strategy

The following **create-auto-scaling-group** command creates an Auto Scaling group that specifies the following:

- The percentage of the group to launch as On-Demand Instances (`0`) and a base number of On-Demand Instances to start with (`1`)
- The instance types to launch in priority order (`c3.large`, `c4.large`, `c5.large`)
- The subnets in which to launch the instances (`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`), each corresponding to a different Availability Zone
- The launch template (`my-launch-template`) and the launch template version (`$Default`)

When Amazon EC2 Auto Scaling attempts to fulfill your On-Demand capacity, it launches the `c3.large` instance type first. The Spot Instances come from the optimal Spot pool in each Availability Zone based on Spot Instance capacity.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "c5.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

Example: Launch Spot Instances using the `lowest-price` allocation strategy diversified over two pools

The following **create-auto-scaling-group** command creates an Auto Scaling group that specifies the following:

- The percentage of the group to launch as On-Demand Instances (`50`) without also specifying a base number of On-Demand Instances to start with
- The instance types to launch in priority order (`c3.large`, `c4.large`, `c5.large`)
- The subnets in which to launch the instances (`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`), each corresponding to a different Availability Zone
- The launch template (`my-launch-template`) and the launch template version (`$Latest`)

When Amazon EC2 Auto Scaling attempts to fulfill your On-Demand capacity, it launches the `c3.large` instance type first. For your Spot capacity, Amazon EC2 Auto Scaling attempts to launch the Spot Instances evenly across the two lowest-priced pools in each Availability Zone.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file.

```json
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "c5.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "lowest-price",
            "SpotInstancePools": 2
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

To verify that the group has launched instances

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 is 3 and that the group has three running instances.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "ServiceLinkedRoleARN": "arn",
            "TargetGroupARNs": [],
            "SuspendedProcesses": [],
            "DesiredCapacity": 3,
            "MixedInstancesPolicy": {
                "InstancesDistribution": {
                    "SpotAllocationStrategy": "lowest-price",
                    "OnDemandPercentageAboveBaseCapacity": 50,
                    "OnDemandAllocationStrategy": "prioritized",
                    "SpotInstancePools": 2,
                    "OnDemandBaseCapacity": 0
                },
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "$Latest",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "Overrides": [
                        {
                            "InstanceType": "c3.large"
                        },
                        {
                            "InstanceType": "c4.large"
                        },
                        {
                            "InstanceType": "c5.large"
                        }
                    ]
                }
            },
            "EnabledMetrics": [],
            "Tags": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0aae8709d49eeba4f",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2b",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0c43f6003841d2d2b",
```

```
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            },
            {
                "ProtectedFromScaleIn": false,
                "AvailabilityZone": "us-west-2c",
                "LaunchTemplate": {
                    "LaunchTemplateName": "my-launch-template",
                    "Version": "1",
                    "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                },
                "InstanceId": "i-0feb4cd6677d39903",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            }
        ],
        "MaxSize": 5,
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "HealthCheckGracePeriod": 0,
        "TerminationPolicies": [
            "Default"
        ],
        "LoadBalancerNames": [],
        "CreatedTime": "2019-02-17T02:29:12.853Z",
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
        ],
        "HealthCheckType": "EC2",
        "NewInstancesProtectedFromScaleIn": false
    }
    ]
}
```

For additional examples, see .

Instance Weighting for Amazon EC2 Auto Scaling

When you configure an Auto Scaling group to launch multiple instance types, you have the option of defining the number of capacity units that each instance contributes to the desired capacity of the group, using *instance weighting*. This allows you to specify the relative weight of each instance type in a way that directly maps to the performance of your application. You can weight your instances to suit your specific application needs, for example, by the cores (vCPUs) or by memory (GiBs).

For example, let's say that you run a compute-intensive application that performs best with at least 8 vCPUs and 15 GiB of RAM. If you use `c5.2xlarge` as your base unit, any of the following EC2 instance types would meet your application needs.

Instance Types Example

Instance type	vCPU	Memory (GiB)
c5.2xlarge	8	16
c5.4xlarge	16	32
c5.12xlarge	48	96
c5.18xlarge	72	144
c5.24xlarge	96	192

By default, all instance types are treated as the same weight. In other words, whether Amazon EC2 Auto Scaling launches a large or small instance type, each instance counts toward the group's desired capacity.

With instance weighting, however, you assign a number value that specifies how many capacity units to associate with each instance type. For example, if the instances are of different sizes, a `c5.2xlarge` instance could have the weight of 2, and a `c5.4xlarge` (which is two times bigger) could have the weight of 4, and so on. Then, when Amazon EC2 Auto Scaling launches instances, their weights count toward your desired capacity.

Price Per Unit Hour

The following table compares the hourly price for Spot Instances in different Availability Zones in US East (N. Virginia, Ohio) with the price for On-Demand Instances in the same Region. The prices shown are example pricing and not current pricing. These are your costs *per instance hour*.

Example: Spot Pricing Per Instance Hour

Instance type	us-east-1a	us-east-1b	us-east-1c	On-Demand pricing
c5.2xlarge	$0.180	$0.191	$0.170	$0.34
c5.4xlarge	$0.341	$0.361	$0.318	$0.68
c5.12xlarge	$0.779	$0.777	$0.777	$2.04
c5.18xlarge	$1.207	$1.475	$1.357	$3.06
c5.24xlarge	$1.555	$1.555	$1.555	$4.08

With instance weighting, you can evaluate your costs based on what you use *per unit hour*. You can determine the price per unit hour by dividing your price for an instance type by the number of units that it represents. For On-Demand Instances, the price *per unit hour* is the same when deploying one instance type as it is when deploying a different size of the same instance type. In contrast, however, the Spot price *per unit hour* varies by Spot pool.

The easiest way to understand how the price *per unit hour* calculation works with weighted instances is with an example. For example, for ease of calculation, let's say you want to launch Spot Instances only in us-east-1a. The *per unit hour price* is captured below.

Example: Spot Price Per Unit Hour Example

Instance type	us-east-1a	Instance weight	Price per unit hour
c5.2xlarge	$0.180	2	$0.090
c5.4xlarge	$0.341	4	$0.085
c5.12xlarge	$0.779	12	$0.065
c5.18xlarge	$1.207	18	$0.067
c5.24xlarge	$1.555	24	$0.065

Considerations

This section discusses the key considerations in implementing instance weighting effectively.

- Start by choosing a few instance types that reflect the actual performance requirements of your application. Then, decide how much each instance type should count toward the desired capacity of

your Auto Scaling group by specifying their weights. The weights apply to current and future instances in the group.

- Be cautious about choosing very large ranges for your weights. For example, we don't recommend specifying a weight of 1 for an instance type when the next larger instance type has a weight of 200. The difference between the smallest and largest weights should also not be extreme. If any of the instance types have too large of a weight difference, this can have a negative effect on ongoing cost-performance optimization.
- The size of the Auto Scaling group is measured in capacity units, and not in instances. For example, if your weights are based on vCPUs, you must specify the desired, minimum, and maximum number of cores you want.
- Set your weights and desired capacity so that the desired capacity is at least two to three times larger than your largest weight.
- If you choose to set your own maximum price for Spot, you must specify a price *per instance hour* that is high enough for your most expensive instance type. Amazon EC2 Auto Scaling provisions Spot Instances if the current Spot price in an Availability Zone is below your maximum price and capacity is available. If the request for Spot Instances cannot be fulfilled in one Spot Instance pool, it keeps trying in other Spot pools to leverage the cost savings of Spot Instances.

With instance weighting, the following new behaviors are introduced:

- Current capacity will either be at the desired capacity or above it. Because Amazon EC2 Auto Scaling wants to provision instances until the desired capacity is totally fulfilled, an overage can happen. For example, suppose that you specify two instance types, `c5.2xlarge` and `c5.12xlarge`, and you assign instance weights of 2 for `c5.2xlarge` and 12 for `c5.12xlarge`. If there are 5 units remaining to fulfill the desired capacity, and Amazon EC2 Auto Scaling provisions a `c5.12xlarge`, the desired capacity is exceeded by 7 units.
- When Amazon EC2 Auto Scaling provisions instances to reach the desired capacity, distributing instances across Availability Zones and respecting the allocation strategies for On-Demand and Spot Instances both take precedence over avoiding overages.
- Amazon EC2 Auto Scaling can overstep the maximum capacity limit to maintain balance across Availability Zones, using your preferred allocation strategies. The hard limit enforced by Amazon EC2 Auto Scaling is a value that is equal to your desired capacity plus your largest weight.

Note the following when adding or modifying weights for existing groups:

- When adding instance weights to an existing Auto Scaling group, you must include any instance types that are already running in the group.
- When modifying existing instance weights, Amazon EC2 Auto Scaling will launch or terminate instances to reach your desired capacity based on the new weights.
- If you remove an instance type, any running instances of that instance type will continue to have their last updated weight values, even though the instance type has been removed.

Add or Modify Weights for Your Auto Scaling Group

You can add weights to an existing Auto Scaling group, or to a new Auto Scaling group as you create it. You can also update an existing Auto Scaling group to define new configuration options (Spot/On-Demand usage, Spot allocation strategy, instance types). If you change how many Spot or On-Demand Instances you want, Amazon EC2 Auto Scaling gradually replaces existing instances to match the new purchase options.

Before creating Auto Scaling groups using instance weighting, we recommend that you become familiar with launching groups with multiple instance types. For more information and additional examples, see .

The following examples show how to use the AWS CLI to add weights when you create Auto Scaling groups, and to add or modify weights for existing Auto Scaling groups. You can configure a variety of parameters in a JSON file, and then reference the JSON file as the sole parameter for your Auto Scaling group.

To add weights to an Auto Scaling group on creation

- Use the **create-auto-scaling-group** command to create a new Auto Scaling group. For example, the following command creates a new Auto Scaling group and adds instance weighting by specifying the following:

  - The percentage of the group to launch as On-Demand Instances (`0`) and a base number of On-Demand Instances to start with (`10`)

  - The allocation strategy for Spot Instances in each Availability Zone (`capacity-optimized`)

  - The instance types to launch in priority order (`m4.16xlarge`, `m5.24xlarge`)

  - The instance weights that correspond to the relative size difference (vCPUs) between instance types (`16`, `24`)

  - The subnets in which to launch the instances (`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`), each corresponding to a different Availability Zone

  - The launch template (`my-launch-template`) and the launch template version (`$Latest`)

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "m4.16xlarge",
                    "WeightedCapacity": "16"
                },
                {
                    "InstanceType": "m5.24xlarge",
                    "WeightedCapacity": "24"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 10,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 160,
    "MaxSize": 720,
    "DesiredCapacity": 480,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

To add or modify weights for an existing Auto Scaling group

- Use the **update-auto-scaling-group** command to add or modify weights. For example, the following command adds weights to instance types in an existing Auto Scaling group by specifying the following:

  - The instance types to launch in priority order (`c5.18xlarge`, `c5.24xlarge`, `c5.2xlarge`, `c5.4xlarge`)

  - The instance weights that correspond to the relative size difference (vCPUs) between instance types (`18, 24, 2, 4`)

  - The new, increased desired capacity, which is larger than the largest weight

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file.

```
{
    "AutoScalingGroupName": "my-existing-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "Overrides": [
                {
                    "InstanceType": "c5.18xlarge",
                    "WeightedCapacity": "18"
                },
                {
                    "InstanceType": "c5.24xlarge",
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceType": "c5.2xlarge",
                    "WeightedCapacity": "2"
                },
                {
                    "InstanceType": "c5.4xlarge",
                    "WeightedCapacity": "4"
                }
            ]
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 100
}
```

To verify the weights for an Auto Scaling group

- Use the following **describe-auto-scaling-groups** command to verify the weights.

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

The following is an example response.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
```

```
            "AutoScalingGroupARN": "arn",
            "MixedInstancesPolicy": {
                "LaunchTemplate": {
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "$Latest"
                    },
                    "Overrides": [
                        {
                            "InstanceType": "m4.16xlarge",
                            "WeightedCapacity": "16"
                        },
                        {
                            "InstanceType": "m5.24xlarge",
                            "WeightedCapacity": "24"
                        }
                    ]
                },
                "InstancesDistribution": {
                    "OnDemandAllocationStrategy": "prioritized",
                    "OnDemandBaseCapacity": 10,
                    "OnDemandPercentageAboveBaseCapacity": 0,
                    "SpotAllocationStrategy": "capacity-optimized"
                }
            },
            "MinSize": 160,
            "MaxSize": 720,
            "DesiredCapacity": 480,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-027327f0ace86f499",
                    "InstanceType": "m5.24xlarge",
                    "AvailabilityZone": "us-west-2a",
                    "LifecycleState": "InService",
                    "HealthStatus": "Healthy",
                    "LaunchTemplate": {
                        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "7"
                    },
                    "ProtectedFromScaleIn": false,
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceId": "i-0ec0d761cc134878d",
                    "InstanceType": "m4.16xlarge",
                    "AvailabilityZone": "us-west-2a",
                    "LifecycleState": "Pending",
                    "HealthStatus": "Healthy",
                    "LaunchTemplate": {
                        "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "7"
                    },
                    "ProtectedFromScaleIn": false,
```

```
                    "WeightedCapacity": "16"
                },
            ...
            }
        ]
}
```

Creating an Auto Scaling Group Using a Launch Template

To configure Amazon EC2 instances that are launched by your Auto Scaling group, you can specify a launch template, a launch configuration, or an EC2 instance. The following procedure demonstrates how to create an Auto Scaling group using a launch template.

With launch templates, you can configure the Auto Scaling group to dynamically choose either the default version or the latest version of the launch template when a scale-out event occurs. For example, you configure your Auto Scaling group to choose the current default version of a launch template. To change the configuration of the EC2 instances to be launched by the group, create or designate a new default version of the launch template. Alternatively, you can choose the specific version of the launch template that the group uses to launch EC2 instances. You can change these selections anytime by updating the group.

Each launch template includes the information that Amazon EC2 needs to launch instances, such as an AMI and instance type. You can create an Auto Scaling group that adheres to the launch template. Or, you can override the instance type in the launch template and combine On-Demand and Spot Instances. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48).

The Auto Scaling group specifies the desired capacity and additional information that Amazon EC2 needs to launch instances, such as the Availability Zones and VPC subnets. You can set capacity to a fixed number of instances, or you can take advantage of automatic scaling to adjust capacity based on actual demand.

Prerequisites

- You must have created a launch template that includes the parameters required to launch an EC2 instance. For information about these parameters and the limitations that apply when creating a launch template for use with an Auto Scaling group, see Creating a Launch Template for an Auto Scaling Group (p. 27).
- You must have IAM permissions to create an Auto Scaling group using a launch template and also to create EC2 resources for the instances. For more information, see Launch Template Support (p. 201).

Amazon EC2 Auto Scaling has changed the Auto Scaling group interface. By default, you're shown the old user interface, but you can switch to the new user interface. This topic contains steps for both.

To create an Auto Scaling group using a launch template (new 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, choose the same AWS Region that you used when you created the launch template.
3. In the navigation pane, choose **Auto Scaling Groups**.
4. Choose **Create an Auto Scaling group**.

5. On the **Choose launch template or configuration** page, do the following:

   a. For **Auto Scaling group name**, enter a name for your Auto Scaling group.

   b. For **Launch Template**, choose an existing launch template.

   c. For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

   d. Verify that your launch template supports all of the options that you are planning to use, and then choose **Next**.

6. On the **Configure settings** page, for **Purchase options and instance types**, choose **Adhere to the launch template** to use the EC2 instance type and purchase option that are specified in the launch template.

7. Under **Network**, for **VPC**, choose the VPC for the security groups that you specified in your launch template. Launching instances using a combination of instance types and purchase options is not supported in EC2-Classic.

8. For **Subnet**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

9. Choose **Next**.

   Or, you can accept the rest of the defaults, and choose **Skip to review**.

10. (Optional) On the **Specify load balancing and health checks** page, configure the following options, and then choose **Next**:

    a. To register your Amazon EC2 instances with a load balancer, choose **Enable load balancing**, and choose an existing load balancer or create a new one. If you use an Application Load Balancer or Network Load Balancer, choose a target group where you register targets by instance ID.

    b. To enable your Elastic Load Balancing (ELB) health checks, for **Health checks**, choose **ELB** under **Health check type**. These health checks are optional when you enable load balancing.

    c. Under **Health check grace period**, enter the amount of time until Amazon EC2 Auto Scaling checks the health of instances after they are put into service. The intention of this setting is to prevent Amazon EC2 Auto Scaling from marking instances as unhealthy and terminating them before they have time to come up. The default is 300 seconds.

11. (Optional) On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**:

    a. For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see Setting Capacity Limits for Your Auto Scaling Group (p. 95).

    b. To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see Target Tracking Scaling Policies (p. 113).

    c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance Scale-In Protection (p. 140).

12. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see Getting Amazon SNS Notifications When Your Auto Scaling Group Scales (p. 162).

13. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

To create an Auto Scaling group using a launch template (old 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, choose the same AWS Region that you used when you created the launch template.

3.   In the navigation pane, choose **Auto Scaling Groups**.

4.   Choose **Create Auto Scaling group**.

5.   Choose **Launch Template**, choose your launch template, and then choose **Next Step**.

6.   On the **Configure Auto Scaling group details** page, for **Group name**, enter a name for your Auto Scaling group.

7.   For **Launch template version**, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.

8.   For **Fleet Composition**, choose **Adhere to the launch template** to use the EC2 instance type and purchase option that are specified in the launch template.

> **Note**
> Alternatively, to launch instances across multiple instance types using both On-Demand and Spot purchase options, choose **Combine purchase options and instances**. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48).

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

10. For **Network**, choose a VPC for your Auto Scaling group.

11. For **Subnet**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

12. (Optional) To register your Amazon EC2 instances with a load balancer, choose **Advanced Details**, choose **Receive traffic from one or more load balancers**, and choose one or more Classic Load Balancers or target groups.

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

14. On the **Configure scaling policies** page, choose one of the following options, and then choose **Next: Configure Notifications**:

- To manually adjust the size of the Auto Scaling group as needed, choose **Keep this group at its initial size**. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

- To automatically adjust the size of the Auto Scaling group based on criteria that you specify, choose **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 113).

15. (Optional) To receive notifications, choose **Add notification**, configure the notification, and then choose **Next: Configure Tags**.

16. (Optional) To add tags, choose **Edit tags**, provide a tag key and value for each tag, and then choose **Review**.

Alternatively, you can add tags later on. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

18. 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 a Launch Configuration

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the subnets for the instances, and the initial number of instances.

> **Important**
> To configure the Amazon EC2 instances, you can specify a launch template, a launch configuration, or an EC2 instance. We recommend that you use a launch template to make sure that you can use the latest features of Amazon EC2. For more information, see Launch Templates (p. 27).

The following procedure demonstrates how to create an Auto Scaling group using a launch configuration. You cannot modify a launch configuration after it is created, but you can replace the launch configuration for an Auto Scaling group. For more information, see Changing the Launch Configuration for an Auto Scaling Group (p. 41).

Prerequisites

Create a launch configuration. For more information, see Creating a Launch Configuration (p. 36).

Amazon EC2 Auto Scaling has changed the Auto Scaling group interface. By default, you're shown the old user interface, but you can switch to the new user interface. This topic contains steps for both.

To create an Auto Scaling group using a launch configuration (new 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, choose the same AWS Region that you used when you created the launch template.
3. In the navigation pane, choose **Auto Scaling Groups**.
4. Choose **Create an Auto Scaling group**.
5. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter a name for your Auto Scaling group.
6. To choose a launch configuration, do the following:

    a. For **Launch Template**, choose **Switch to launch configuration**.
    b. For **Launch configuration**, choose an existing launch configuration.
    c. Verify that your launch configuration supports all of the options that you are planning to use, and then choose **Next**.

7. On the **Configure settings** page, under **Network**, for **VPC**, choose the VPC for the security groups that you specified in your launch configuration. Launching instances using a combination of instance types and purchase options is not supported in EC2-Classic.
8. For **Subnet**, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).
9. Choose **Next**.

    Or, you can accept the rest of the defaults, and choose **Skip to review**.
10. (Optional) On the **Specify load balancing and health checks** page, configure the following options, and then choose **Next**:

    a. To register your Amazon EC2 instances with a load balancer, choose **Enable load balancing**, and choose an existing load balancer or create a new one. If you use an Application Load Balancer or Network Load Balancer, choose a target group where you register targets by instance ID.

b. To enable your Elastic Load Balancing (ELB) health checks, for **Health checks**, choose **ELB** under **Health check type**. These health checks are optional when you enable load balancing.

c. Under **Health check grace period**, enter the amount of time until Amazon EC2 Auto Scaling checks the health of instances after they are put into service. The intention of this setting is to prevent Amazon EC2 Auto Scaling from marking instances as unhealthy and terminating them before they have time to come up. The default is 300 seconds.

11. (Optional) On the **Configure group size and scaling policies** page, configure the following options, and then choose **Next**:

   a. For **Desired capacity**, enter the initial number of instances to launch. When you change this number to a value outside of the minimum or maximum capacity limits, you must update the values of **Minimum capacity** or **Maximum capacity**. For more information, see Setting Capacity Limits for Your Auto Scaling Group (p. 95).

   b. To automatically scale the size of the Auto Scaling group, choose **Target tracking scaling policy** and follow the directions. For more information, see Target Tracking Scaling Policies (p. 113).

   c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance Scale-In Protection (p. 140).

12. (Optional) To receive notifications, for **Add notification**, configure the notification, and then choose **Next**. For more information, see Getting Amazon SNS Notifications When Your Auto Scaling Group Scales (p. 162).

13. (Optional) To add tags, choose **Add tag**, provide a tag key and value for each tag, and then choose **Next**. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

To create an Auto Scaling group using a launch configuration (old 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, choose the same AWS 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, choose **Launch Configuration**, choose an existing 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**, enter a name for your Auto Scaling group.

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

   c. For **Network**, choose a VPC for your Auto Scaling group.

   d. For **Subnet**, choose one or more subnets in your VPC. Use subnets in multiple Availability Zones for high availability. For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

   e. (Optional) To register your Amazon EC2 instances with a load balancer, choose **Advanced Details**, choose **Receive traffic from one or more load balancers**, and choose one or more Classic Load Balancers or target groups.

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

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

- To manually adjust the size of the Auto Scaling group as needed, choose **Keep this group at its initial size**. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

- To automatically adjust the size of the Auto Scaling group based on criteria that you specify, choose **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 113).

8. (Optional) To receive notifications, choose **Add notification**, configure the notification, and then choose **Next: Configure Tags**.

9. (Optional) To add tags, choose **Edit tags**, provide a tag key and value for each tag, and then choose **Review**.

   Alternatively, you can add tags later on. For more information, see Tagging Auto Scaling Groups and Instances (p. 72).

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

11. 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

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the subnets for the instances, and the initial number of instances.

To configure Amazon EC2 instances, you can specify a launch configuration, a launch template, or an EC2 instance. The following procedure demonstrates how to create an Auto Scaling group using an EC2 instance. To use a launch configuration or a launch template, see Creating an Auto Scaling Group Using a Launch Configuration (p. 67) or Creating an Auto Scaling Group Using a Launch Template (p. 64).

When you create an Auto Scaling group using an EC2 instance, Amazon EC2 Auto Scaling 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 from the specified instance, such as AMI ID, instance type, and Availability Zone.

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.
- If the identified instance is registered with one or more load balancers, the information about the load balancer is not copied to the load balancer or target group 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 to create the Auto Scaling group.
- The instance is not a member of another Auto Scaling group.
- The instance is in the `running` state.
- The AMI used to launch the instance must still exist.

Contents

Create an Auto Scaling Group from an EC2 Instance (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. 98).

Create an Auto Scaling Group from an EC2 Instance (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
```

Use the following **describe-auto-scaling-groups** command to describe the Auto Scaling group.

```
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 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-configurations** 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 using 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 add 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 that 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, choose **Keep this group at its initial size**. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

    - To automatically adjust the size of the Auto Scaling group based on criteria that you specify, choose **Use scaling policies to adjust the capacity of this group** and follow the directions. For more information, see Configure Scaling Policies (p. 113).

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

Tags help you to categorize your Auto Scaling groups in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each Auto Scaling group. Additionally, you can propagate the tags from the Auto Scaling group to the Amazon EC2 instances it launches. Tagging your 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*.

You can also control which IAM users and groups in your account have permission to create, edit, or delete tags. For more information, see Control Which Tag Keys and Tag Values Can Be Used (p. 195). Keep in mind, however, that a policy that restricts your users from performing a tagging operation on an Auto Scaling group does not prevent them from manually changing the tags on the instances after they have launched. For information about IAM policies for tagging (or untagging) Amazon EC2 resources, see Example: Tagging Resources in the *Amazon EC2 User Guide for Linux Instances*.

Tags are not propagated to Amazon EBS volumes. To add tags to Amazon EBS volumes, specify the tags in a launch template but use caution when configuring instance tags in your launch template. If the launch template specifies an instance tag with a key that is also specified for the Auto Scaling group, Amazon EC2 Auto Scaling overrides the value of that instance tag with the value specified by the Auto Scaling group. For information about specifying tags in a launch template, see Creating a Launch Template for an Auto Scaling Group (p. 27).

Contents

Tag Restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 50.
- The maximum number of tags that you can add or remove using a single call is 25.
- The maximum key length is 128 Unicode characters.
- The maximum value length is 256 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 toward your limit of tags per Auto Scaling group.

Tagging Lifecycle

If you have opted to propagate tags to your Amazon EC2 instances, the tags are managed as follows:

- When an Auto Scaling group launches instances, it adds tags to the instances during resource creation rather than after the resource is created.
- The Auto Scaling group automatically adds a tag to the instances with a key of `aws:autoscaling:groupName` and a value of the name of the Auto Scaling group.
- When you attach existing instances, the Auto Scaling group adds the tags to the instances, overwriting any existing tags with the same tag key. In addition, it adds a tag with a key of `aws:autoscaling:groupName` and a value of the name of the Auto Scaling group.
- When you detach an instance from an Auto Scaling group, it removes only the `aws:autoscaling:groupName` tag.
- When you scale in manually or the Auto Scaling group automatically scales in, it removes all tags from the instances that are terminating.

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 the 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 (Console)

Use the Amazon EC2 console to:

- Add tags to new Auto Scaling groups when you create them
- Add, modify, or delete tags for existing Auto Scaling groups

To tag an Auto Scaling group on creation

When you use the Amazon EC2 console to create an Auto Scaling group, you can specify tag keys and values on the **Configure Tags** page of the Create Auto Scaling Group wizard. To propagate a tag to the instances launched in the Auto Scaling group, make sure that you keep the **Tag New Instances** option for that tag selected. Otherwise, you can deselect it.

To add or modify tags for an existing Auto Scaling group

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

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

3. Select the check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group you selected.

4. On the **Details** tab, under **Tags**, choose **Edit**. (Old console: On the **Tags** tab, choose **Add/Edit tags**.)

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 **Update**. (Old console: Choose **Save**.)

Add or Modify Tags (AWS CLI)

The following examples show how to use the AWS CLI to add tags when you create Auto Scaling groups, and to add or modify tags for existing Auto Scaling groups.

To tag an Auto Scaling group on creation

- Use the **create-auto-scaling-group** command to create a new Auto Scaling group and add a tag, for example, `env=prod`, to the Auto Scaling group. The tag is also added to any instances launched in the Auto Scaling group.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --tags Key=env,Value=prod,PropagateAtLaunch=true
```

To create or modify tags for an existing Auto Scaling group

- Use the **create-or-update-tags** command to create or modify a tag. For example, the following command adds the `Name=my-asg` and `cost-center=cc123` tags. The tags are also added to any instances launched in the Auto Scaling group after this change. If a tag with either key already exists, the existing tag is replaced. The Amazon EC2 console associates the display name for each instance with the name that is specified for the `Name` key (case-sensitive).

```
aws autoscaling create-or-update-tags \
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-
asg,PropagateAtLaunch=true \
  ResourceId=my-asg,ResourceType=auto-scaling-group,Key=cost-
center,Value=cc123,PropagateAtLaunch=true
```

To list all tags for an Auto Scaling group

- 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": "prod",
            "Key": "env"
        }
    ]
}
```

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

Delete Tags

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

Contents

Delete Tags (Console)

To delete a tag (new 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 the check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group you selected.
4. On the **Details** tab, under **Tags**, choose **Edit**.
5. Choose **Remove** next to the tag.
6. Choose **Update**.

To delete a tag (old 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. Choose an existing group from the list.
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 (AWS CLI)

Use the **delete-tags** command to delete a tag. For example, the following command deletes a tag with a key of `env`.

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

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

Elastic Load Balancing and Amazon EC2 Auto Scaling

A load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group. When an instance is added to your group, it needs to register with the load balancer or no traffic is routed to it. When an instance is removed from your group, it must deregister from the load balancer or traffic continues to be routed to it.

When you use your Elastic Load Balancing load balancer with an Auto Scaling group, it's not necessary to register your EC2 instances with the load balancer or target group. With Elastic Load Balancing, instances that are launched by your Auto Scaling group are automatically registered with the load balancer or target group, and instances that are terminated by your Auto Scaling group are automatically deregistered from the load balancer or target group.

You can also configure Elastic Load Balancing health checks to monitor the health of registered instances so that the load balancer or target group only routes traffic to the healthy instances.

Contents

Elastic Load Balancing Types

Elastic Load Balancing provides three types of load balancers that can be used with your Auto Scaling group: Classic Load Balancers, Application Load Balancers, and Network Load Balancers. With Classic Load Balancers, instances are registered with the load balancer. With Application Load Balancers and Network Load Balancers, instances are registered as targets with a target group.

Classic Load Balancer

> Routes and load balances either at the transport layer (TCP/SSL), or at the application layer (HTTP/HTTPS). A Classic Load Balancer supports either EC2-Classic or a VPC.

Application Load Balancer

> Routes and load balances at the application layer (HTTP/HTTPS), and supports path-based routing. An Application Load Balancer can route requests to ports on one or more registered targets, such as EC2 instances, in your virtual private cloud (VPC).
>
> > **Note**
> > The Application Load Balancer target groups must have a target type of `instance`. For more information, see Target Type in the *User Guide for Application Load Balancers*.

Network Load Balancer

> Routes and load balances at the transport layer (TCP/UDP Layer-4), based on address information extracted from the TCP packet header, not from packet content. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.
>
> > **Note**
> > The Network Load Balancer target groups must have a target type of `instance`. For more information, see Target Type in the *User Guide for Network Load Balancers*.

To learn more about Elastic Load Balancing, see the following topics:

- What Is Elastic Load Balancing?
- What Is a Classic Load Balancer?
- What Is an Application Load Balancer?
- What Is a Network Load Balancer?

Attaching a Load Balancer to Your Auto Scaling Group

This topic describes how to attach your Elastic Load Balancing load balancer to an existing Auto Scaling group. To attach your load balancer to your Auto Scaling group when you create the group, see Tutorial: Set Up a Scaled and Load-Balanced Application (p. 19).

Amazon EC2 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.

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, Amazon EC2 Auto Scaling can terminate and replace any instances that are reported as unhealthy. 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. Amazon EC2 Auto Scaling doesn't terminate and replace the instances.

When you detach a load balancer, it enters the `Removing` state while deregistering the instances in the group. 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. By default, connection draining is enabled for Application Load Balancers but must be enabled for Classic Load Balancers. For more information, see Connection Draining in the *User Guide for Classic Load Balancers*.

Contents

- Prerequisites (p. 78)
- Attach a Load Balancer (Console) (p. 78)
- Attach a Load Balancer (AWS CLI) (p. 79)

Prerequisites

Before you begin, create an Application Load Balancer or Network Load Balancer in the same AWS Region as the Auto Scaling group. We recommend the new load balancers, but you can still use a Classic Load Balancer if it supports the features you're looking for. To learn more about the different types of load balancers, see Elastic Load Balancing Types (p. 77).

(Optional) To configure your Auto Scaling group to use Elastic Load Balancing health checks, see Adding Elastic Load Balancing Health Checks to an Auto Scaling Group (p. 80).

> **Note**
> Amazon EC2 Auto Scaling has changed the Auto Scaling group interface. By default, you're shown the old user interface, but you can switch to the new user interface. This topic contains steps for both.

Attach a Load Balancer (Console)

Use the following procedure to attach a load balancer to an existing Auto Scaling group.

To attach a load balancer to a group (new 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 the check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group that you selected.
4. On the **Details** tab, under **Load balancing**, choose **Edit**.
5. Under **Load balancing**, do one of the following:

   a. [Application/Network Load Balancers] For **Choose a target group for your load balancer**, choose your target group.

b. [Classic Load Balancers] For **Choose a load balancer**, choose your load balancer.

6. Choose **Update**.

To attach a load balancer to a group (old 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. Choose an existing group from the list.
4. On the **Details** tab, choose **Edit**.
5. Do one of the following:

   a. [Classic Load Balancers] For **Classic Load Balancers**, choose your load balancer.
   b. [Application/Network Load Balancers] For **Target Groups**, choose 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 (new 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 the check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group that you selected.
4. On the **Details** tab, under **Load balancing**, choose **Edit**.
5. Under **Load balancing**, do one of the following:

   a. [Application/Network Load Balancers] For **Choose a target group for your load balancer**, choose the delete icon (X) next to the target group.
   b. [Classic Load Balancers] For **Choose a load balancer**, choose the delete icon (X) next to the load balancer.

6. Choose **Update**.

To detach a load balancer from a group (old 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. Choose an existing group from the list.
4. On the **Details** tab, choose **Edit**.
5. Do one of the following:

   a. [Classic Load Balancers] For **Classic Load Balancers**, remove the load balancer.
   b. [Application/Network Load Balancers] For **Target Groups**, remove the target group.

6. Choose **Save**.

Attach a Load Balancer (AWS CLI)

To attach a target group for an Application Load Balancer or Network Load Balancer

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 target group for an Application Load Balancer or Network Load Balancer

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
```

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 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
```

Adding Elastic Load Balancing Health Checks to an Auto Scaling Group

The default health checks for an Auto Scaling group are EC2 status checks only. If an instance fails these status checks, the Auto Scaling group considers the instance unhealthy and replaces it. For more information, see Health Checks for Auto Scaling Instances (p. 160).

You can attach one or more target groups (Application Load Balancers and Network Load Balancers), one or more load balancers (Classic Load Balancers), or both to your Auto Scaling group. However, by default, the group does not consider an instance unhealthy and replace it if it fails the Elastic Load Balancing health checks.

To ensure that the group can determine an instance's health based on additional tests provided by the load balancer, you can optionally configure the Auto Scaling group to use Elastic Load Balancing health checks. The load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called health checks.

To learn more about Elastic Load Balancing health checks, see the following topics:

- Configure Health Checks for Your Classic Load Balancer in the *User Guide for Classic Load Balancers*
- Health Checks for Your Target Groups in the *User Guide for Application Load Balancers*
- Health Checks for Your Target Groups in the *User Guide for Network Load Balancers*

If you configure the Auto Scaling group to use Elastic Load Balancing health checks, it considers the instance unhealthy if it fails either the EC2 status checks or the Elastic Load Balancing health checks. If you attach multiple load balancer target groups or Classic Load Balancers to the group, all of them must

report that the instance is healthy in order for it to consider the instance healthy. If any one of them reports an instance as unhealthy, the Auto Scaling group replaces the instance, even if other ones report it as healthy.

The following procedures show how to add Elastic Load Balancing health checks to your Auto Scaling group.

Contents

- Adding Health Checks (Console) (p. 81)
- Adding Health Checks (AWS CLI) (p. 81)

Adding Health Checks (Console)

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

To add health checks

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 check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group you selected.
4. On the **Details** tab, under **Health checks**, choose **Edit**. (Old console: On the **Details** tab, choose **Edit**.)
5. For **Health check type**, select **Enable ELB health checks**. (Old console: Select **ELB**.)
6. For **Health check grace period**, enter 300.
7. Choose **Update**. (Old console: Choose **Save**.)
8. On the **Instance management** tab, under **Instances**, you can view the health status of instances. (Old console: The **Instances** tab is where you can view the health status of instances.) The **Health Status** column displays the results of the newly added health checks.

Adding Health Checks (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 zones. Incoming traffic is distributed equally across all Availability Zones enabled for your load balancer.

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

When one Availability Zone becomes unhealthy or unavailable, Amazon EC2 Auto Scaling launches new instances in an unaffected zone. When the unhealthy Availability Zone returns to a healthy state, Amazon EC2 Auto Scaling automatically redistributes the application instances evenly across all of the zones for your Auto Scaling group. Amazon EC2 Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Amazon EC2 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 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 zones.

Contents

Add an Availability Zone (Console)

Use the following procedure to expand your Auto Scaling group to an additional subnet.

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Auto Scaling Groups**.
3. Select the check box next to an existing group.

   A new pane appears below the **Auto Scaling groups** pane, showing information about the group you selected.
4. On the **Details** tab, under **Network**, choose **Edit**. (Old console: On the **Details** tab, choose **Edit**.)
5. In **Subnets**, choose the subnet corresponding to the Availability Zone.
6. Choose **Update**. (Old console: Choose **Save**.)
7. To update the Availability Zones for your load balancer so that it shares the same zones as your Auto Scaling group, complete the following steps:

   a. On the navigation pane, choose **Load Balancers**.
   b. Choose your load balancer.
   c. 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, choose 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**.
   d. Choose **Save**.

Add an Availability Zone (AWS CLI)

The commands that you 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
```

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. This topic describes how to launch only Spot Instances in your Auto Scaling group by specifying settings in a *launch configuration* or *launch template*, rather than in the Auto Scaling group itself.

> **Important**
> You can specify the same settings that are used to launch Spot Instances as part of the settings of an Auto Scaling group. When you specify the settings as part of the Auto Scaling group, you can specify additional options. For example, you can specify whether to launch only Spot Instances, or a combination of both On-Demand Instances and Spot Instances. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options (p. 48).

Before launching Spot Instances using Amazon EC2 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*.

When you create a launch configuration or launch template to launch Spot Instances instead of On-Demand Instances, keep the following considerations in mind:

- **Setting your maximum price.** You set the maximum price you are willing to pay as part of the launch configuration or launch template. If the Spot price is within your maximum price, whether your request is fulfilled depends on Spot Instance capacity. You pay only the Spot price for the Spot Instances that you launch. If the price for Spot Instances rises above your maximum price for a running instance in your Auto Scaling group, Amazon EC2 terminates your instance. For more information, see Pricing and Savings in the *Amazon EC2 User Guide for Linux Instances*.

- **Changing your maximum price.** You must create a launch configuration or launch template version with the new price. With a new launch configuration, you must associate it with your Auto Scaling group. With a launch template, you can configure the Auto Scaling group to use the default template or the latest version of the template. That way, it is automatically associated with the Auto Scaling group. The existing instances continue to run as long as the maximum price specified in the launch configuration or launch template used for those instances is higher than the current Spot price.

- **Maintaining your Spot Instances.** When your Spot Instance is terminated, the Auto Scaling group attempts to launch a replacement instance to maintain the desired capacity for the group. If your maximum price is higher than the Spot price, then it launches a Spot Instance. Otherwise (or if the request is unsuccessful), it keeps trying.

- **Balancing across Availability Zones.** If you specify multiple Availability Zones, Amazon EC2 Auto Scaling distributes the Spot requests across the specified zones. If your maximum price is too low in one Availability Zone for any requests to be fulfilled, Amazon EC2 Auto Scaling checks whether requests were fulfilled in the other zones. If so, Amazon EC2 Auto Scaling cancels the requests that failed and redistributes them across the Availability Zones that have requests fulfilled. If the price in an Availability Zone with no fulfilled requests drops enough that future requests succeed, Amazon EC2 Auto Scaling rebalances across all of the Availability Zones. For more information, see Rebalancing Activities (p. 7).

- **Spot Instance termination.** Amazon EC2 Auto Scaling can terminate or replace Spot Instances in the same way that it can terminate or replace On-Demand Instances. For more information, see Controlling Which Auto Scaling Instances Terminate During Scale In (p. 137).

- **Spot interruption notices.** You can use Spot Instance interruption notices to monitor the status of your Spot Instances. For example, you can set up a rule in Amazon CloudWatch Events that automatically sends the EC2 Spot two-minute warning to an Amazon SNS topic, an AWS Lambda function, or another target. For more information, see Spot Instance Interruption Notices in the *Amazon EC2 User Guide for Linux Instances* and the Amazon CloudWatch Events User Guide.

Getting Recommendations for an Instance Type

AWS provides Amazon EC2 instance recommendations to help you improve performance, save money, or both, by using features powered by AWS Compute Optimizer. You can use these recommendations to decide whether to move to a new instance type.

To make recommendations, Compute Optimizer analyzes your existing instance specifications and recent metric history. The compiled data is then used to recommend which Amazon EC2 instance types are best optimized to handle the existing performance workload. Recommendations are returned along with per-hour instance pricing.

> **Note**
> To get recommendations from Compute Optimizer, you must first opt in to Compute Optimizer. For more information, see Getting Started with AWS AWS Compute Optimizer in the *AWS Compute Optimizer User Guide*.

Contents

Limitations

Compute Optimizer currently generates recommendations for M, C, R, T, and X instance types. Other instance types are not considered by Compute Optimizer. When you use other instance types, they are excluded from the recommendations.

Compute Optimizer currently generates recommendations for Auto Scaling groups that have the same values for desired, minimum, and maximum capacity, and that are configured to launch a single instance type.

Findings

Compute Optimizer classifies its findings for Auto Scaling groups as follows:

- **Not optimized** – An Auto Scaling group is considered not optimized when Compute Optimizer has identified a recommendation that can provide better performance for your workload.
- **Optimized** – An Auto Scaling group is considered optimized when Compute Optimizer determines that the group is correctly provisioned to run your workload, based on the chosen instance type. For optimized resources, Compute Optimizer might sometimes recommend a new generation instance type.
- **None** – There are no recommendations for this Auto Scaling group. This might occur if you've been opted in to Compute Optimizer for less than 12 hours, or when the Auto Scaling group has been running for less than 30 hours, or when the Auto Scaling group or instance type is not supported by Compute Optimizer. For more information, see Limitations (p. 85) in the previous section.

Viewing Recommendations

After you opt in to Compute Optimizer, you can view the findings and recommendations that it generates for your Auto Scaling groups. If you recently opted in, recommendations might not be available for up to 12 hours.

To view recommendations generated for an Auto Scaling group

1. Open the Compute Optimizer console at https://console.aws.amazon.com/compute-optimizer/.

   The Dashboard page opens.

2. Choose **View recommendations for all Auto Scaling groups**.

3. Select your Auto Scaling group.

4. Choose **View detail**.

   The view changes to display up to three different instance recommendations in a preconfigured view, based on default table settings. It also provides recent CloudWatch metric data (average CPU utilization, average network in, and average network out) for the Auto Scaling group.

Determine whether you want to use one of the recommendations. Decide whether to optimize for performance improvement, for cost reduction, or for a combination of these two.

To change the instance type in your Auto Scaling group, you must create a new launch template or launch configuration. If you update the Auto Scaling group to use the new launch template or launch configuration and increase the desired capacity of the group, new instances will be launched with the new instance type. Or, you can terminate existing instances in the Auto Scaling group to force a replacement instance to launch that uses your new launch template or launch configuration. With CloudFormation, you also have the option of automating the update.

Considerations for Evaluating the Recommendations

Before moving to a new instance type, consider the following:

- The recommendations don't forecast your usage. Recommendations are based on your historical usage over the most recent 14-day time period. Be sure to choose an instance type that is expected to meet your future usage needs.
- Focus on the graphed metrics to determine whether actual usage is lower than instance capacity. You can also view metric data (average, peak, percentile) in CloudWatch to further evaluate your EC2 instance recommendations. For example, notice how CPU percentage metrics change during the day and whether there are peaks that need to be accommodated. For more information, see Viewing Available Metrics in the *Amazon CloudWatch User Guide*.
- Compute Optimizer might supply recommendations for burstable performance instances, which are T3, T3a, and T2 instances. If you periodically burst above your baseline, make sure that you can continue to do so based on the vCPUs of the new instance type. For more information, see CPU Credits and Baseline Performance for Burstable Performance Instances in the *Amazon EC2 User Guide for Linux Instances*.
- If you've purchased a Reserved Instance, your On-Demand Instance might be billed as a Reserved Instance. Before you change your current instance type, first evaluate the impact on Reserved Instance utilization and coverage.
- Consider conversions to newer generation instances, where possible.
- When migrating to a different instance family, make sure the current instance type and the new instance type are compatible, for example, in terms of virtualization, architecture, or network type. For more information, see Compatibility for Resizing Instances in the *Amazon EC2 User Guide for Linux Instances*.
- Finally, consider the performance risk rating that's provided for each recommendation. Performance risk indicates the amount of effort you might need to spend in order to validate whether the recommended instance type meets the performance requirements of your workload. We also recommend rigorous load and performance testing before and after making any changes.

Additional resources

In addition to the topics on this page, see the following resources:

- Amazon EC2 Instance Types
- AWS Compute Optimizer User Guide

Replacing Auto Scaling Instances Based on Maximum Instance Lifetime

When you use the AWS Management Console to update an Auto Scaling group, or when you use the AWS CLI or AWS SDKs to create or update an Auto Scaling group, you can set the optional maximum instance lifetime parameter. The maximum instance lifetime feature does the work of replacing instances that have been in service for the maximum amount of time allowed. For example, this feature supports common compliance use cases, such as being required to replace your instances on a schedule due to internal security policies or external compliance controls. This topic describes the key aspects of this feature and how to configure it for your Auto Scaling group.

The maximum instance lifetime specifies the maximum amount of time (in seconds) that an instance can be in service. The maximum duration applies to all current and future instances in the group. As an instance approaches its maximum duration, it is terminated and replaced, and cannot be used again.

When configuring the maximum instance lifetime for your Auto Scaling group, you must specify a value of at least 604,800 seconds (7 days). To clear a previously set value, specify a new value of 0.

Note that instances are not guaranteed to be replaced only at the end of their maximum duration. In some situations, Amazon EC2 Auto Scaling might need to start replacing instances immediately after you configure the maximum instance lifetime parameter. The intention of this more aggressive behavior is to avoid replacing all instances at the same time.

Depending on the maximum duration specified and the size of the Auto Scaling group, the rate of replacement may vary. In general, Amazon EC2 Auto Scaling replaces instances one at a time, with a pause in between replacements. However, the rate of replacement will be higher when there is not enough time to replace each instance individually based on the maximum duration that you specified. In this case, Amazon EC2 Auto Scaling will replace several instances at once, by up to 10 percent of the current capacity of your Auto Scaling group at a time.

To manage the rate of replacement, you can do the following:

- Set the maximum instance lifetime limit to a longer period of time to space out the replacements. This is helpful for groups that have a large number of instances to replace.
- Add extra time between certain replacements by using instance protection to temporarily prevent individual instances in your Auto Scaling group from being replaced. When you're ready to replace these instances, remove instance protection from each individual instance. For more information, see Instance Scale-In Protection (p. 140).

To configure maximum instance lifetime (console)

Create the Auto Scaling group in the usual way. After creating the Auto Scaling group, edit the group to specify the maximum instance lifetime.

To configure maximum instance lifetime (AWS CLI)

When specifying the maximum instance lifetime using the AWS CLI, you can apply this limit to an existing Auto Scaling group. You can also apply this limit to a new Auto Scaling group as you create it.

For new Auto Scaling groups, use the **create-auto-scaling-group** command.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

The following is an example `config.json` file that shows a maximum instance lifetime of `2592000` seconds (30 days).

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Latest"
    },
    "MinSize": 1,
    "MaxSize": 5,
    "MaxInstanceLifetime": 2592000,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

For existing Auto Scaling groups, use the **update-auto-scaling-group** command.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --max-
instance-lifetime 2592000
```

To verify the maximum instance lifetime for an Auto Scaling group

Use the **describe-auto-scaling-groups** command.

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

The following is an example response.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-04d180b9d5fc578fc",
                    "InstanceType": "t2.small",
                    "AvailabilityZone": "us-west-2b",
                    "LifecycleState": "Pending",
```

```
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "7"
            },
            "ProtectedFromScaleIn": false
        }
    ],
    "CreatedTime": "2019-11-14T22:56:15.487Z",
    "SuspendedProcesses": [],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "EnabledMetrics": [],
    "Tags": [],
    "TerminationPolicies": [
        "Default"
    ],
    "NewInstancesProtectedFromScaleIn": false,
    "ServiceLinkedRoleARN": "arn",
    "MaxInstanceLifetime": 2592000
        }
    ]
}
```

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

To merge separate single-zone Auto Scaling groups into a single group spanning multiple Availability Zones, rezone one of the single-zone groups into a multi-zone group. Then, delete the other groups. This 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 (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`. 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-launch-config",
            "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 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 instances are terminated. Deleting an instance also deletes any associated logs or data, and any volumes on the instance. If do not want to terminate one or more instances, you can detach them before you delete the Auto Scaling group. If the group has scaling policies, deleting the group deletes the policies, the underlying alarm actions, and any alarm that no longer has an associated action.

To delete your Auto Scaling group (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 the check box next to your Auto Scaling group and choose **Delete** (Old console: choose **Actions**, **Delete**).
4. When prompted for confirmation, choose **Delete**.

   A loading icon in the **Name** column indicates that the Auto Scaling group is being deleted. The **Desired**, **Min**, and **Max** columns show 0 instances for the Auto Scaling group. It takes a few minutes to terminate the instance and delete the group. Refresh the list to see the current state.

To delete your Auto Scaling group (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
```

If the group has instances or scaling activities in progress, use the **delete-auto-scaling-group** command with the `--force-delete` option. This will also terminate the Amazon EC2 instances.

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

(Optional) Delete the Launch Configuration

You can skip this step to keep the launch configuration for future use.

To delete the launch configuration (console)

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

To delete the launch configuration (AWS CLI)

Use the following **delete-launch-configuration** command.

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

(Optional) Delete the Launch Template

You can delete your launch template or just one version of your launch template. When you delete a launch template, all its versions are deleted.

You can skip this step to keep the launch template for future use.

To delete your launch template (console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Launch Templates**.
3. Select your launch template and then do one of the following:

   - Choose **Actions**, **Delete template**. When prompted for confirmation, choose **Delete launch template**.
   - Choose **Actions**, **Delete template version**. Select the version to delete and choose **Delete launch template version**.

To delete the launch template (AWS CLI)

Use the following **delete-launch-template** command to delete your template and all its versions.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

Alternatively, you can use the **delete-launch-template-versions** command to delete a specific version of a launch template.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --
versions 1
```

(Optional) Delete the Load Balancer

Skip this step if your Auto Scaling group is not associated with an Elastic Load Balancing load balancer, or if you want to keep the load balancer for future use.

To delete your load balancer (console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Choose the load balancer and choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

To delete your target group (console)

1. On the navigation pane, choose **Target Groups**.
2. Choose the target group and choose **Actions**, **Delete**.
3. When prompted for confirmation, choose **Yes**.

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

For Application Load Balancers and Network Load Balancers, use the following **delete-load-balancer** and **delete-target-group** commands.

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

For Classic Load Balancers, use the following **delete-load-balancer** command.

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

(Optional) Delete CloudWatch Alarms

To delete any CloudWatch alarms associated with your Auto Scaling group, complete the following steps.

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

> **Note**
> Deleting an Auto Scaling group automatically deletes the CloudWatch alarms that Amazon EC2 Auto Scaling manages for a target tracking scaling policy.

To delete the CloudWatch alarms (console)

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

To delete the CloudWatch alarms (AWS CLI)

Use the **delete-alarms** command. You can delete one or more alarms at a time. For example, use the following command to delete the `Step-Scaling-AlarmHigh-AddCapacity` and `Step-Scaling-AlarmLow-RemoveCapacity` alarms.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-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 an Auto Scaling group to either launch or terminate Amazon EC2 instances.

Amazon EC2 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. Keep the following considerations in mind:

- What role should Amazon EC2 Auto Scaling play in your application's architecture? It's common to think about automatic scaling primarily as a way to increase and decrease capacity, but it's also useful for maintaining a steady number of servers.
- What cost constraints are important to you? Because Amazon EC2 Auto Scaling uses EC2 instances, you only pay for the resources that you use. Knowing your cost constraints helps you decide when to scale your applications, and by how much.
- What metrics are important to your application? Amazon CloudWatch supports a number of different metrics that you can use with your Auto Scaling group.

Contents

Scaling Options

Amazon EC2 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 specified number of running instances at all times. To maintain the current instance levels, Amazon EC2 Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When Amazon EC2 Auto Scaling finds an unhealthy instance, it terminates that instance and launches a new one. For more information, see Maintaining a Fixed Number of Instances in Your Auto Scaling Group (p. 96).

Scale manually

Manual scaling is the most basic way to scale your resources, where you specify only the change in the maximum, minimum, or desired capacity of your Auto Scaling group. Amazon EC2 Auto Scaling manages

the process of creating or terminating instances to maintain the updated capacity. For more information, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

Scale based on a schedule

Scaling by schedule means that scaling actions are performed automatically as a function of time and date. This is useful when you know exactly when to increase or decrease the number of instances in your group, simply because the need arises on a predictable schedule. For more information, see Scheduled Scaling for Amazon EC2 Auto Scaling (p. 105).

Scale based on demand

A more advanced way to scale your resources, using scaling policies, lets you define parameters that control the scaling process. For example, let's say that you have a web application that currently runs on two instances and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This method is useful for scaling in response to changing conditions, when you don't know when those conditions will change. You can set up Amazon EC2 Auto Scaling to respond for you. For more information, see Dynamic Scaling for Amazon EC2 Auto Scaling (p. 108).

Use predictive scaling

You can also use Amazon EC2 Auto Scaling in combination with AWS Auto Scaling to scale resources across multiple services. AWS Auto Scaling can help you maintain optimal availability and performance by combining predictive scaling and dynamic scaling (proactive and reactive approaches, respectively) to scale your Amazon EC2 capacity faster. For more information, see the AWS Auto Scaling User Guide.

Setting Capacity Limits for Your Auto Scaling Group

You configure the size of your Auto Scaling group by setting the minimum, maximum, and desired capacity. The minimum and maximum capacity are required to create an Auto Scaling group, while the desired capacity is optional. If you do not define your desired capacity up front, it defaults to your minimum capacity.

> **Note**
> By default, the minimum, maximum, and desired capacity are set to one instance when you create an Auto Scaling group from the console. If you change the desired capacity, the capacity that you specify will be the total number of instances launched right after creating your Auto Scaling group.

An Auto Scaling group is elastic as long as it has different values for minimum and maximum capacity. All requests to change the Auto Scaling group's desired capacity (either by manual scaling or automatic scaling) must fall within these limits.

If you choose to automatically scale your group, the **maximum** limit lets Amazon EC2 Auto Scaling scale out the number of instances as needed to handle an increase in demand. The **minimum** limit helps ensure that you always have a certain number of instances running at all times.

These limits also apply when you manually scale your Auto Scaling group, such as when you want to turn off automatic scaling and have the group run at a fixed size, either temporarily or permanently.

To access capacity settings in 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 to view it.

4. On the **Details** tab, view or change the current settings for minimum, maximum, and desired capacity.

Maintaining a Fixed Number of Instances in Your Auto Scaling Group

After you have created your Auto Scaling group, the Auto Scaling group starts by launching enough EC2 instances to meet its minimum capacity (or its desired capacity, if specified).

If a fixed number of instances are needed, this can be achieved by setting the same value for minimum, maximum, and desired capacity. If there are no other scaling conditions attached to the Auto Scaling group, the group maintains this number of running instances even if an instance becomes unhealthy.

To maintain the same number of instances, Amazon EC2 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. If you stop or terminate a running instance, the instance is considered to be unhealthy and is replaced. For more information about health check replacements, see Health Checks for Auto Scaling Instances (p. 160).

To manually scale the Auto Scaling group, you can adjust the desired capacity to update the number of instances that Amazon EC2 Auto Scaling attempts to maintain. Before you can adjust the desired capacity to a value outside of the minimum and maximum capacity range, you must update these limits.

Manual Scaling for Amazon EC2 Auto Scaling

At any time, you can change the size of an existing Auto Scaling group manually. You can either update the desired capacity of the Auto Scaling group, or update the instances that are attached to the Auto Scaling group. Manually scaling your group can be useful when automatic scaling is not needed or when you need to hold capacity at a fixed number of instances.

Changing the Size of Your Auto Scaling Group (Console)

When you change the desired capacity of your Auto Scaling group, Amazon EC2 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 capacity**, increase the desired capacity by one. For example, if the current value is 1, enter 2.

    The desired capacity must be less than or equal to the maximum size of the group. If your new value for **Desired capacity** is greater than **Maximum capacity**, you must update **Maximum capacity**.
6. When you are finished, choose **Save changes**.

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** tab, in **Activity history**, the **Status** column shows the current status of your instance. Use the refresh button until you see the status of your instance change to **Successful**. This indicates that your Auto Scaling group has successfully launched a new instance.

    > **Note**
    > If the instance fails to launch, you can find troubleshooting tips in Troubleshooting Amazon EC2 Auto Scaling (p. 214).

2. On the **Instance management** tab, in **Instances**, 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.

Changing the Size of Your Auto Scaling Group (AWS CLI)

When you change the size of your Auto Scaling group, Amazon EC2 Auto Scaling manages the process of launching or terminating instances to maintain the new group size. The default behavior is not to wait for the default cooldown period to complete, but you can override the default and wait for the cooldown period to complete. For more information, see Scaling Cooldowns for Amazon EC2 Auto Scaling (p. 133).

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

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
```

If you choose to honor the default cooldown period for your Auto Scaling group, you must specify the `--honor-cooldown` option as shown in the following example.

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

To verify the size of your Auto Scaling group

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
```

The following is example output, with details about the group and instances launched.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "ServiceLinkedRoleARN": "arn",
```

```
            "TargetGroupARNs": [],
            "SuspendedProcesses": [],
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0c20ac468fa3049e8",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            "MaxSize": 5,
            "VPCZoneIdentifier": "subnet-c87f2be0",
            "HealthCheckGracePeriod": 300,
            "TerminationPolicies": [
                "Default"
            ],
            "CreatedTime": "2019-03-18T23:30:42.611Z",
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "HealthCheckType": "EC2",
            "NewInstancesProtectedFromScaleIn": false,
            "DesiredCapacity": 2
        }
    ]
}
```

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

Attach EC2 Instances to Your Auto Scaling Group

Amazon EC2 Auto Scaling provides you with an option to enable automatic 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 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 launched into one of the Availability Zones defined in your 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 load balancer must both be in the same VPC.

When you attach instances, the desired capacity of the group increases 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.

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 (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

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

5. (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

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**, and then select an instance.

4. Choose **Actions**, **Instance Settings**, **Attach to Auto Scaling Group**.

5. On the **Attach to Auto Scaling Group** page, select **an existing Auto Scaling group**, select an instance, and then choose **Attach**.

6. If the instance doesn't meet the criteria, you get an error message with the details. For example, the instance might not be in the same Availability Zone as the Auto Scaling group. Choose **Close** and try again with an instance that meets the criteria.

Attaching an Instance (AWS CLI)

To attach an instance to an Auto Scaling group

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 that the group has two running instances.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "ServiceLinkedRoleARN": "arn",
            "TargetGroupARNs": [],
            "SuspendedProcesses": [],
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
```

```
            },
            "InstanceId": "i-0c20ac468fa3049e8",
            "HealthStatus": "Healthy",
            "LifecycleState": "InService"
          }
        ],
        "MaxSize": 5,
        "VPCZoneIdentifier": "subnet-c87f2be0",
        "HealthCheckGracePeriod": 300,
        "TerminationPolicies": [
            "Default"
        ],
        "CreatedTime": "2019-03-18T23:30:42.611Z",
        "AvailabilityZones": [
            "us-west-2a"
        ],
        "HealthCheckType": "EC2",
        "NewInstancesProtectedFromScaleIn": false,
        "DesiredCapacity": 2
      }
    ]
}
```

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

```
aws autoscaling attach-instances --instance-ids i-0787762faf1c28619 --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 instance (to a new capacity of 3), and that there is a new instance, i-0787762faf1c28619.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "ServiceLinkedRoleARN": "arn",
            "TargetGroupARNs": [],
            "SuspendedProcesses": [],
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
```

```
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0c20ac468fa3049e8",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0787762faf1c28619",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            "MaxSize": 5,
            "VPCZoneIdentifier": "subnet-c87f2be0",
            "HealthCheckGracePeriod": 300,
            "TerminationPolicies": [
                "Default"
            ],
            "CreatedTime": "2019-03-18T23:30:42.611Z",
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "HealthCheckType": "EC2",
            "NewInstancesProtectedFromScaleIn": false,
            "DesiredCapacity": 3
        }
    ]
}
```

Detach EC2 Instances from Your Auto Scaling Group

You can remove (detach) 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 group. For more information, see Attach EC2 Instances to Your Auto Scaling Group (p. 98).

- Test an Auto Scaling group by creating it using existing instances running your application. You can 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 that you are detaching. If you choose not to decrement the capacity, Amazon EC2 Auto Scaling launches new instances to replace the ones that you detach. If you decrement the capacity but detach multiple instances from the same Availability Zone, Amazon EC2 Auto

Scaling can rebalance the Availability Zones unless you suspend the `AZRebalance` process. For more information, see Scaling Processes (p. 155).

If the number of instances that you are detaching decreases the size of the Auto Scaling group below its minimum capacity, you must decrement the minimum capacity for the group before you can detach the instances.

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, Amazon EC2 Auto Scaling waits for in-flight requests to complete.

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 (Console)

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

To detach an instance from an existing 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 **Instance management** tab, in **Instances**, select an instance and choose **Actions**, **Detach**. (Old console: The **Instances** tab is where you can detach the instance.)
5. In the **Detach instance** dialog box, select the check box to launch a replacement instance, or leave it unchecked to decrement the desired capacity. Choose **Detach instance**.

Detaching Instances (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

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 four running instances.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
```

```
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-05b4f7d5be44822a6",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
    },
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0c20ac468fa3049e8",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
    },
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0787762faf1c28619",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
    },
    {
        "ProtectedFromScaleIn": false,
        "AvailabilityZone": "us-west-2a",
        "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1",
            "LaunchTemplateId": "lt-050555ad16a3f9c7f"
        },
        "InstanceId": "i-0f280a4c58d319a8a",
        "AutoScalingGroupName": "my-asg",
        "HealthStatus": "HEALTHY",
        "LifecycleState": "InService"
    }
  ]
}
```

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

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --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 three running instances.

```
{
    "AutoScalingInstances": [
```

```
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0c20ac468fa3049e8",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0787762faf1c28619",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0f280a4c58d319a8a",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        }
    ]
}
```

Scheduled Scaling for Amazon EC2 Auto Scaling

Scheduled scaling allows you to set your own scaling schedule. For example, let's say that every week the traffic to your web application starts to increase on Wednesday, remains high on Thursday, and starts to decrease on Friday. You can plan your scaling actions based on the predictable traffic patterns of your web application. Scaling actions are performed automatically as a function of time and date.

> **Note**
> For scaling based on predictable load changes, you can also use the predictive scaling feature of AWS Auto Scaling. For more information, see the AWS Auto Scaling User Guide.

To configure your Auto Scaling group to scale based on a schedule, you create a scheduled action. The scheduled action tells Amazon EC2 Auto Scaling to perform a scaling action at specified times. To create a scheduled scaling action, you specify the start time when the scaling action should take effect, and the new minimum, maximum, and desired sizes for the scaling action. At the specified time, Amazon EC2 Auto Scaling updates the group with the values for minimum, maximum, and desired size that are specified by the scaling action.

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

Considerations

When you create a scheduled action, keep the following in mind:

- A scheduled action sets the minimum, maximum, and desired sizes to what is specified by the scheduled action at the time specified by the scheduled action. It does not keep track of old values and return to the older values after the end time.

- A scheduled action generally executes within seconds. However, the action might be delayed for up to two minutes from the scheduled start time. Because actions within an Auto Scaling group are executed in the order that they are specified, scheduled actions with scheduled start times close to each other can take longer to execute.

- The order of execution for scheduled actions is guaranteed within the same group, but not for scheduled actions across groups.

- 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.

- You can create a maximum of 125 scheduled actions per Auto Scaling group.

- A scheduled action does not persist in your account once it has reached its end time.

- You can temporarily disable scheduled scaling without deleting your scheduled actions. For more information, see Suspending and Resuming Scaling Processes (p. 154).

- Cooldown periods are not supported.

- You can also schedule scaling actions for resources beyond Amazon EC2. For more information, see Scheduled Scaling in the *Application Auto Scaling User Guide*.

Create and Manage Scheduled Actions (Console)

You can create scheduled actions that scale one time only or that scale on a recurring schedule using the console. Complete the following procedure to create a scheduled action to scale your Auto Scaling group.

To create a scheduled action for an 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 **Automatic scaling** tab, in **Scheduled actions**, choose **Create scheduled action**. (Old console: The **Scheduled Actions** tab is where you can create a scheduled action.)

5. In the **Create scheduled action** dialog box, do the following:

   - For **Name**, enter a name for the scheduled action.

   - Specify the size of the group using at least one of the following values: **Min**, **Max**, or **Desired capacity**.

   - Choose an option for **Recurrence**. If you choose **Once**, the action is performed at the specified time. If you choose **Cron**, enter a cron expression that specifies when to perform the action, in UTC. If you choose an option that begins with **Every**, the cron expression is created for you.

   - If you chose **Once** for **Recurrence**, specify the date and time for the action to run in **Start time**.

   - For recurrent actions, you can specify values for both **Start time** and **End time**. If you specify a start time, the earliest time that the action will be performed is at this time. If you specify an end time, the action stops repeating after this time.

6. 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 **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action. (Old console: The **Scheduled Actions** tab is where you can select the scheduled action.)
5. Choose **Actions**, **Edit**.
6. In the **Edit scheduled action** dialog box, do the following:

   - Update the size of the group as needed using **Min**, **Max**, or **Desired capacity**.
   - Update the specified recurrence as needed.
   - Update the start and end time as needed.
7. Choose **Save**.

Delete a Scheduled Action

When you no longer need a scheduled action, you can delete it.

To delete 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 **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action. (Old console: The **Scheduled Actions** tab is where you can select the scheduled action.)
5. Choose **Actions**, **Delete**.
6. When prompted for confirmation, choose **Yes, Delete**.

Create and Manage Scheduled Actions (AWS CLI)

You can create and update scheduled actions that scale one time only or that scale on a recurring schedule using the **put-scheduled-update-group-action** command.

To scale one time only

You can specify a one-time schedule to automatically scale your Auto Scaling group at a certain date and time, in UTC.

- To decrease the number of running instances in your Auto Scaling group at a specific time, use the following command. At the date and time specified for `--start-time`, if the group currently has more than 1 instance, the group scales in to 1 instance.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-
action \
  --auto-scaling-group-name my-asg --start-time "2019-05-13T08:00:00Z" --desired-
capacity 1
```

- To increase the number of running instances in your Auto Scaling group at a specific time, use the following command. At the date and time specified for `--start-time`, if the group currently has fewer than 3 instances, the group scales out to 3 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-
action \
  --auto-scaling-group-name my-asg --start-time "2019-05-12T08:00:00Z" --desired-
capacity 3
```

To scale on a recurring schedule

You can specify a recurrence schedule, in UTC, using the Unix cron syntax format. This format consists of five fields separated by white spaces: [Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]. For more information about this format, see Crontab.

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 my-recurring-
action \
  --auto-scaling-group-name my-asg --recurrence "30 0 1 1,6,12 *" --desired-capacity 3
```

Delete a Scheduled Action

To delete a scheduled action

Use the following **delete-scheduled-action** command.

```
aws autoscaling delete-scheduled-action --scheduled-action-name my-recurring-action
```

Dynamic Scaling for Amazon EC2 Auto Scaling

When you configure dynamic scaling, you define how to scale the capacity of your Auto Scaling group in response to changing demand.

For example, let's say that you have a web application that currently runs on two instances, and you want the CPU utilization of the Auto Scaling group to stay at around 50 percent when the load on the application changes. This gives you extra capacity to handle traffic spikes without maintaining an excessive number of idle resources.

You can configure your Auto Scaling group to scale dynamically to meet this need by creating a scaling policy. Amazon EC2 Auto Scaling can then scale out your group (add more instances) to deal with high demand at peak times, and scale in your group (run fewer instances) to reduce costs during periods of low utilization.

Contents

How Scaling Policies Work

A scaling policy instructs Amazon EC2 Auto Scaling to track a specific CloudWatch metric, and it defines what action to take when the associated CloudWatch alarm is in ALARM. The metrics that are used to trigger an alarm are an aggregation of metrics coming from all of the instances in the Auto Scaling group. (For example, let's say you have an Auto Scaling group with two instances where one instance is at 60 percent CPU and the other is at 40 percent CPU. On average, they are at 50 percent CPU.) When the policy is in effect, Amazon EC2 Auto Scaling adjusts the group's desired capacity up or down when the alarm is triggered.

When a scaling policy is executed, if the capacity calculation produces a number outside of the minimum and maximum size range of the group, Amazon EC2 Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits. Capacity is measured in one of two ways: using the same units that you chose when you set the desired capacity in terms of instances, or using capacity units (if instance weighting (p. 58) is applied).

- Example 1: An Auto Scaling group has a maximum capacity of 3, a current capacity of 2, and a scaling policy that adds 3 instances. When executing this scaling policy, Amazon EC2 Auto Scaling adds only 1 instance to the group to prevent the group from exceeding its maximum size.
- Example 2: An Auto Scaling group has a minimum capacity of 2, a current capacity of 3, and a scaling policy that removes 2 instances. When executing this scaling policy, Amazon EC2 Auto Scaling removes only 1 instance from the group to prevent the group from becoming less than its minimum size.

When the desired capacity reaches the maximum size limit, scaling out stops. If demand drops and capacity decreases, Amazon EC2 Auto Scaling can scale out again.

The exception is when you use instance weighting. In this case, Amazon EC2 Auto Scaling can scale out above the maximum size limit, but only by up to your maximum instance weight. Its intention is to get as close to the new desired capacity as possible but still adhere to the allocation strategies that are specified for the group. The allocation strategies determine which instance types to launch. The weights determine how many capacity units each instance contributes to the desired capacity of the group based on its instance type.

- Example 3: An Auto Scaling group has a maximum capacity of 12, a current capacity of 10, and a scaling policy that adds 5 capacity units. Instance types have one of three weights assigned: 1, 4, or 6. When executing the scaling policy, Amazon EC2 Auto Scaling chooses to launch an instance type with a weight of 6 based on the allocation strategy. The result of this scale-out event is a group with a desired capacity of 12 and a current capacity of 16.

Scaling Policy Types

Amazon EC2 Auto Scaling supports the following types of scaling policies:

- **Target tracking scaling**—Increase or decrease the current capacity of the group based on a target value for a specific metric. This is similar to the way that your thermostat maintains the temperature of your home—you select a temperature and the thermostat does the rest.
- **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**—Increase or decrease the current capacity of the group based on a single scaling adjustment.

If you are scaling based on a utilization metric that increases or decreases proportionally to the number of instances in an Auto Scaling group, we recommend that you use target tracking scaling policies. Otherwise, we recommend that you use step scaling policies.

Multiple Scaling Policies

In most cases, a target tracking scaling policy is sufficient to configure your Auto Scaling group to scale out and scale in automatically. A target tracking scaling policy allows you to select a desired outcome and have the Auto Scaling group add and remove instances as needed to achieve that outcome.

For an advanced scaling configuration, your Auto Scaling group can have more than one scaling policy. For example, you can define one or more target tracking scaling policies, one or more step scaling policies, or both. This provides greater flexibility to cover multiple scenarios.

To illustrate how multiple scaling policies work together, consider an application that uses an Auto Scaling group and an Amazon SQS queue to send requests to a single EC2 instance. To help ensure that the application performs at optimum levels, there are two policies that control when the Auto Scaling group should scale out. One is a target tracking policy that uses a custom metric to add and remove capacity based on the number of SQS messages in the queue. The other is a step scaling policy that uses the Amazon CloudWatch `CPUUtilization` metric to add capacity when the instance exceeds 90 percent utilization for a specified length of time.

When there are multiple policies in force at the same time, there's a chance that each policy could instruct the Auto Scaling group to scale out (or in) at the same time. For example, it's possible that the `CPUUtilization` metric spikes and triggers the CloudWatch alarm at the same time that the SQS custom metric spikes and triggers the custom metric alarm.

When these situations occur, Amazon EC2 Auto Scaling chooses the policy that provides the largest capacity for both scale out and scale in. Suppose, for example, that the policy for `CPUUtilization` launches one instance, while the policy for the SQS queue launches two instances. If the scale-out criteria for both policies are met at the same time, Amazon EC2 Auto Scaling gives precedence to the SQS queue policy. This results in the Auto Scaling group launching two instances.

The approach of giving precedence to the policy that provides the largest capacity applies even when the policies use different criteria for scaling in. For example, if one policy terminates three instances, another policy decreases the number of instances by 25 percent, and the group has eight instances at the time of scale in, Amazon EC2 Auto Scaling gives precedence to the policy that provides the largest number of instances for the group. This results in the Auto Scaling group terminating two instances (25 percent of 8 = 2). The intention is to prevent Amazon EC2 Auto Scaling from removing too many instances.

We recommend caution, however, when using target tracking scaling policies with step scaling policies because conflicts between these policies can cause undesirable behavior. For example, if the step scaling policy initiates a scale-in activity before the target tracking policy is ready to scale in, the scale-in activity will not be blocked. After the scale-in activity completes, the target tracking policy could instruct the group to scale out again.

Target Tracking Scaling Policies for Amazon EC2 Auto Scaling

With target tracking scaling policies, you select a scaling metric and set a target value. Amazon EC2 Auto Scaling creates and manages the CloudWatch alarms that trigger the scaling policy and calculates the scaling adjustment based on the metric and the target value. The scaling policy adds or removes capacity as required to keep the metric at, or close to, the specified target value. In addition to keeping the metric close to the target value, a target tracking scaling policy also adjusts to the changes in the metric due to a changing load pattern.

For example, you can use target tracking scaling to:

- Configure a target tracking scaling policy to keep the average aggregate CPU utilization of your Auto Scaling group at 40 percent.
- Configure a target tracking scaling policy to keep the request count per target of your Elastic Load Balancing target group at 1000 for your Auto Scaling group.

We recommend that you scale on Amazon EC2 instance metrics with a 1-minute frequency because that ensures a faster response to utilization changes. Scaling on metrics with a 5-minute frequency can result in slower response times and scaling on stale metric data. By default, Amazon EC2 instances are enabled for basic monitoring, which means metric data for instances is available at 5-minute intervals. You can enable detailed monitoring to get metric data for instances at 1-minute frequency. For more information, see Configure Monitoring for Auto Scaling Instances (p. 176).

Considerations

Keep the following considerations in mind:

- A target tracking scaling policy assumes that it should scale out your Auto Scaling group when the specified metric is above the target value. You cannot use a target tracking scaling policy to scale out your Auto Scaling group when the specified metric is below the target value.
- You might see gaps between the target value and the actual metric data points. This is because we act conservatively by rounding up or down when determining how many instances to add or remove. This prevents us from adding an insufficient number of instances or removing too many instances. However, for smaller Auto Scaling groups with fewer instances, the utilization of the group might seem far from the target value. For example, let's say that you set a target value of 50 percent for CPU utilization and your Auto Scaling group then exceeds the target. We might determine that adding 1.5 instances will decrease the CPU utilization to close to 50 percent. Because it is not possible to add 1.5 instances, we round up and add two instances. This might decrease the CPU utilization to a value below 50 percent, but it ensures that your application has enough resources to support it. Similarly, if we determine that removing 1.5 instances increases your CPU utilization to above 50 percent, we remove just one instance.
- For larger Auto Scaling groups with more instances, the utilization is spread over a larger number of instances, in which case adding or removing instances causes less of a gap between the target value and the actual metric data points.
- To ensure application availability, the Auto Scaling group scales out proportionally to the metric as fast as it can, but scales in more gradually.
- An Auto Scaling group can have multiple scaling policies in force at the same time. For more information, see Multiple Scaling Policies (p. 110).
- You can have multiple target tracking scaling policies for an Auto Scaling group, provided that each of them uses a different metric. The intention of Amazon EC2 Auto Scaling is to always prioritize availability, so its behavior differs depending on whether the target tracking policies are ready for scale out or scale in. It will scale out the Auto Scaling group if any of the target tracking policies are ready for scale out, but will scale in only if all of the target tracking policies (with the scale-in portion enabled) are ready to scale in.
- You can disable the scale-in portion of a target tracking scaling policy. This feature provides you with the flexibility to scale in your Auto Scaling group using a different method. For example, you can use a different scaling policy type for scale in while using a target tracking scaling policy for scale out.
- Do not edit or delete the CloudWatch alarms that are configured for the target tracking scaling policy. CloudWatch alarms that are associated with your target tracking scaling policies are deleted automatically when you delete the scaling policies.
- You can also use target tracking scaling for resources beyond Amazon EC2. For more information, see Target Tracking Scaling Policies in the *Application Auto Scaling User Guide*.

Choosing Metrics

You can use the Amazon EC2 console to apply a target tracking scaling policy based on a predefined metric. Alternatively, you can use the AWS CLI or AWS SDKs to apply a scaling policy based on a predefined or customized metric. The following predefined metrics are available:

- `ASGAverageCPUUtilization`—Average CPU utilization of the Auto Scaling group.
- `ASGAverageNetworkIn`—Average number of bytes received on all network interfaces by the Auto Scaling group.
- `ASGAverageNetworkOut`—Average number of bytes sent out on all network interfaces by the Auto Scaling group.
- `ALBRequestCountPerTarget`—Number of requests completed per target in an Application Load Balancer target group.

You can choose other available Amazon CloudWatch metrics by specifying a customized metric.

Keep the following in mind when choosing a metric:

- Not all metrics work for target tracking. This can be important when you are specifying a customized metric. The metric must be a valid utilization metric and describe how busy an instance is. The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group. That's so the metric data can be used to proportionally scale out or in the number of instances. For example, the CPU utilization of an Auto Scaling group works (that is, the Amazon EC2 metric `CPUUtilization` with the metric dimension `AutoScalingGroupName`), if the load on the Auto Scaling group is distributed across the instances.
- The following metrics do not work for target tracking:
  - The number of requests received by the load balancer fronting the Auto Scaling group (that is, the Elastic Load Balancing metric `RequestCount`). The number of requests received by the load balancer doesn't change based on the utilization of the Auto Scaling group.
  - Load balancer request latency (that is, the Elastic Load Balancing metric `Latency`). Request latency can increase based on increasing utilization, but doesn't necessarily change proportionally.
  - The CloudWatch SQS queue metric `ApproximateNumberOfMessagesVisible`. The number of messages in a queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. However, a customized metric that measures the number of messages in the queue per EC2 instance in the Auto Scaling group can work. For more information, see .
- A target tracking scaling policy does not scale in your Auto Scaling group when the specified metric has insufficient data, unless you use the `ALBRequestCountPerTarget` metric. This works because the `ALBRequestCountPerTarget` metric emits zeros for periods with no associated data, and the target tracking policy requires metric data to interpret a low utilization trend. To have your Auto Scaling group scale in to 0 instances when no requests are routed to the target group, the group's minimum capacity must be set to 0.

Instance Warm-up

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, we do not consider instances that are warming up as part of the current capacity of the group. This ensures that we don't add more instances than you need.

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

A scale-in activity can't start while a scale-out activity is in progress.

Create a Target Tracking Scaling Policy for an Auto Scaling Group (Console)

To create a target tracking scaling policy

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 and verify that the minimum and maximum size limits are appropriately set. For example, if your group is already at its maximum size, specify a new maximum in order to scale out. Amazon EC2 Auto Scaling does not scale your group below the minimum capacity or above the maximum capacity. To update your group, choose **Edit**, specify the minimum and maximum size, and then choose **Save**.
4. On the **Automatic scaling** tab, in **Scaling policies**, choose **Add policy**. (Old console: The **Scaling Policies** tab is where you can add the policy.)
5. To define a policy, do the following:

   a. For **Policy type**, leave the default of **Target tracking scaling**.
   b. Specify a name for the policy.
   c. Choose a metric for **Metric type**.
   d. Specify a **Target value** for the metric.
   e. (Optional) Specify an instance warm-up value for **Instances need**. This allows you to control the time until a newly launched instance can contribute to the CloudWatch metrics.
   f. (Optional) Select **Disable scale in to create only a scale-out policy**. This allows you to create a separate scale-in policy of a different type if wanted.
6. Choose **Create**.

Create a Target Tracking Scaling Policy (AWS CLI)

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

Tasks

Step 1: Create an Auto Scaling Group

Use the **create-auto-scaling-group** command to create an Auto Scaling group named `my-asg` using the launch configuration `my-launch-config`. 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-launch-config \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --max-size 5 --min-size 1
```

Step 2: Create a Target Tracking Scaling Policy

You can create a target tracking scaling policy that tells the Auto Scaling group to increase and decrease the number of running EC2 instances in the group dynamically when the load on the application changes.

Example: target tracking configuration file

The following is an example of a target tracking configuration file, which you should save as `config.json`.

```
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
}
```

Alternatively, you can customize the metric used for scaling by creating a customized metric specification and adding values for each parameter from CloudWatch. The following is an example target tracking configuration that keeps the average utilization of the specified metric at 40 percent.

```
{
    "TargetValue":40.0,
    "CustomizedMetricSpecification":{
        "MetricName":"MyUtilizationMetric",
        "Namespace":"MyNamespace",
        "Dimensions":[
            {
                "Name":"MyOptionalMetricDimensionName",
                "Value":"MyOptionalMetricDimensionValue"
            }
        ],
        "Statistic":"Average",
        "Unit":"Percent"
    }
}
```

Example: cpu40-target-tracking-scaling-policy

Use the **put-scaling-policy** command, along with the `config.json` file that you created previously, to create a scaling policy named `cpu40-target-tracking-scaling-policy` that keeps the average CPU utilization of the Auto Scaling group at 40 percent.

```
aws autoscaling put-scaling-policy --policy-name cpu40-target-tracking-scaling-policy \
   --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
   --target-tracking-configuration file://config.json
```

Step and Simple Scaling Policies for Amazon EC2 Auto Scaling

With step scaling and simple scaling, you choose scaling metrics and threshold values for the CloudWatch alarms that trigger the scaling process. You also define how your Auto Scaling group should be scaled when a threshold is in breach for a specified number of evaluation periods.

We strongly recommend that you use a target tracking scaling policy to scale on a metric like average CPU utilization or the `RequestCountPerTarget` metric from the Application Load Balancer. Metrics that decrease when capacity increases and increase when capacity decreases can be used to proportionally scale out or in the number of instances using target tracking. This helps ensure that Amazon EC2 Auto Scaling follows the demand curve for your applications closely. For more information, see Target Tracking Scaling Policies (p. 110). You still have the option to use step scaling as an additional policy for a more advanced configuration. For example, you can configure a more aggressive response when demand reaches a certain level.

Differences Between Step Scaling Policies and Simple Scaling Policies

Step scaling policies and simple scaling policies are two of the dynamic scaling options available for you to use. Both require you to create CloudWatch alarms for the scaling policies. Both require you to specify the high and low thresholds for the alarms. Both require you to define whether to add or remove instances, and how many, or set the group to an exact size.

The main difference between the policy types is the step adjustments that you get with step scaling policies. When *step adjustments* are applied, and they increase or decrease the current capacity of your Auto Scaling group, the adjustments vary based on the size of the alarm breach.

In most cases, step scaling policies are a better choice than simple scaling policies, even if you have only a single scaling adjustment.

The main issue with simple scaling is that 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 responding to additional alarms. Cooldown periods help to prevent the initiation of additional scaling activities before the effects of previous activities are visible.

In contrast, with step scaling the policy can continue 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 Amazon EC2 Auto Scaling as it receives the alarm messages.

Amazon EC2 Auto Scaling originally supported only simple scaling policies. If you created your scaling policy before target tracking and step policies were introduced, your policy is treated as a simple scaling policy.

Step Adjustments

When you create a step scaling policy, you specify one or more step adjustments that automatically scale the number of instances dynamically based on the size of the alarm breach. Each step adjustment specifies the following:

- A lower bound for the metric value
- An upper bound for the metric value
- The amount by which to scale, based on the scaling adjustment type

CloudWatch aggregates metric data points based on the statistic for the metric that is associated with your CloudWatch alarm. When the alarm is breached, the appropriate scaling policy is triggered. Amazon EC2 Auto Scaling applies the aggregation type to the most recent metric data points from CloudWatch (as opposed to the raw metric data). It compares this aggregated metric value against the upper and lower bounds defined by the step adjustments to determine which step adjustment to perform.

You specify the upper and lower bounds relative to the breach threshold. For example, let's say that you have an Auto Scaling group that has both a current capacity and a desired capacity of 10. You have a CloudWatch alarm with a breach threshold of 50 percent and scale-out and scale-in policies. You have a set of step adjustments with an adjustment type of `PercentChangeInCapacity` (or **Percent of group** in the console) for each policy:

Example: Step adjustments for scale-out policy

Lower Bound	Upper Bound	Adjustment
0	10	0
10	20	10

Lower Bound	Upper Bound	Adjustment
20	null	30

Example: Step adjustments for scale-in policy

Lower Bound	Upper Bound	Adjustment
-10	0	0
-20	-10	-10
null	-20	-30

This creates the following scaling configuration.

```
Metric value

-infinity          30%     40%            60%      70%                infinity
----------------------------------------------------------------------
         -30%        | -10% | Unchanged  | +10% |      +30%
----------------------------------------------------------------------
```

The following points summarize the behavior of the scaling configuration in relation to the desired and current capacity of the group:

- The desired and current capacity is maintained while the aggregated metric value is greater than 40 and less than 60.
- If the metric value gets to 60, the desired capacity of the group increases 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, the current capacity of the group increases to 11 instances. If the metric value rises to 70 even after this increase in capacity, the desired capacity of the group increases by another 3 instances, to 14 instances. This is 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, the desired capacity of the group decreases 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, the desired capacity of the group decreases by another 3 instances, to 10 instances. This is 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).

When you specify the step adjustments for your scaling policy, note the following:

- If you are using the AWS Management Console, you specify the upper and lower bounds as absolute values. If you are using the AWS CLI or AWS SDKs, you specify the upper and lower bounds relative to the breach threshold.
- The ranges of your step adjustments can't overlap or have a gap.
- Only 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.
- Only 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.

Scaling Adjustment Types

You can define a scaling policy that performs the optimal scaling action, based on the scaling adjustment type that you choose. You can specify the adjustment type as a percentage of the current capacity of your Auto Scaling group, or in capacity units. Normally a capacity unit means one instance, unless you are using the instance weighting feature.

Amazon EC2 Auto Scaling supports the following adjustment types for step scaling and simple scaling:

- `ChangeInCapacity` — Increment or decrement the current capacity of the group by the specified value. A positive value increases the capacity and a negative adjustment value decreases the capacity. For example: If the current capacity of the group is 3 and the adjustment is 5, then when this policy is performed, we add 5 capacity units to the capacity for a total of 8 capacity units.
- `ExactCapacity` — Change the current capacity of the group to the specified value. Specify a positive value with this adjustment type. For example: If the current capacity of the group is 3 and the adjustment is 5, then when this policy is performed, we change the capacity to 5 capacity units.
- `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. For example: If the current capacity is 10 and the adjustment is 10 percent, then when this policy is performed, we add 1 capacity unit to the capacity for a total of 11 capacity units.

    Note
    If the resulting value is not an integer, it is rounded 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`.

With `PercentChangeInCapacity`, you can also specify the minimum number of instances to scale using the `MinAdjustmentMagnitude` parameter. For example, suppose that you create a policy that adds 25 percent and you specify a minimum increment of 2 instances. If you have an Auto Scaling group with 4 instances and the scaling policy is executed, 25 percent of 4 is 1 instance. However, because you specified a minimum increment of 2, there are 2 instances added.

When is used, the effect of setting the `MinAdjustmentMagnitude` parameter to a non-zero value changes. The value is in capacity units. To set the minimum number of instances to scale, set this parameter to a value that is at least as large as your maximum instance weight.

If you are using instance weighting, keep in mind that the current capacity of your Auto Scaling group can exceed the desired capacity as needed. If your absolute number to decrement, or the amount that the percentage says to decrement, is less than the difference between current and desired capacity, no scaling action is taken. You must take these behaviors into account when you look at the outcome of a scaling policy when an alarm is triggered. For example, suppose that the desired capacity is 30 and the current capacity is 32. When the alarm is triggered, if the scaling policy decrements the desired capacity by 1, then no scaling action is taken.

Instance Warm-up

If you are creating a step policy, 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.

Using the example in the Step Adjustments 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 1 instance is added (10 percent of 10 instances). However, the desired capacity of the group is already 11 instances, so the scaling policy does not increase the desired capacity further. If the metric gets to 70 while the new instance is still warming up, we should add 3 instances (30 percent of 10 instances). However, the desired capacity of the group is already 11, so we add only 2 instances, for a new desired capacity of 13 instances.

While scaling out, we do 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.

While scaling in, we consider instances that are terminating as part of the current capacity of the group. Therefore, we don't remove more instances from the Auto Scaling group than necessary.
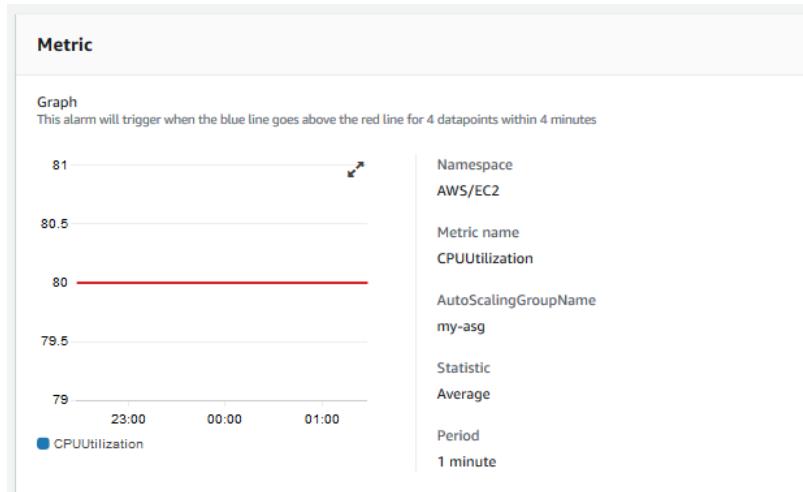
A scale-in activity can't start while a scale-out activity is in progress.

Create Step Scaling Policies for an Auto Scaling Group (Console)

The following procedure shows you how to use the Amazon EC2 Auto Scaling console to create two step 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.

Your step scaling policy 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 Amazon EC2 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 are sent to CloudWatch from the services in AWS that you're using. Or, you can create and monitor your own 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.

Example: CloudWatch alarm that monitors the average CPU utilization metric

To create a CloudWatch alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**.

3. Choose **Create alarm**.

4. On the **Specify metric and conditions** page, choose **Select metric**.

5. Choose **EC2**, **By Auto Scaling Group**, select the `CPU Utilization` metric for your group, and then choose **Select metric**.

    The **Specify metric and conditions** page appears, showing a graph and other information about the metric you selected.

6. For **Period**, choose the evaluation period for the alarm, for example, 1 minute. When evaluating the alarm, each period is aggregated into one data point.

    > **Note**
    > A shorter period creates a more sensitive alarm.

7. For **Conditions**, define the alarm by defining the threshold condition. For example, you can define a threshold to trigger the alarm whenever the value of the metric is greater than or equal to 80 percent.

8. Under **Additional configuration**, for **Datapoints to alarm**, specify how many datapoints (evaluation periods) must be in the ALARM state to trigger the alarm, for example, 1 evaluation period or 2 out of 3 evaluation periods. This creates an alarm that goes to ALARM state if that many consecutive periods are breaching. For more information, see Evaluating an Alarm in the *Amazon CloudWatch User Guide*.

9. For **Missing data treatment**, choose one of the options (or leave the default of **Treat missing data as missing**). For more information, see Configuring How CloudWatch Alarms Treat Missing Data in the *Amazon CloudWatch User Guide*.

10. Choose **Next**.

11. (Optional) To receive notification of a scaling event, for **Notification**, you can choose or create the Amazon SNS topic you want to use to receive notifications. Otherwise, you can delete the notification now and add one later as needed.

12. Choose **Next**.

13. Under **Add a description**, enter a name (for example, `Step-Scaling-AlarmHigh-AddCapacity`) and description (for example, `ASG CPU above 80%`) for the alarm and choose **Next**.

14. Choose **Create alarm**.

To create a step scaling policy for scale out

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 and verify that the minimum and maximum size limits are appropriately set. For example, if your group is already at its maximum size, you need to specify a new maximum in order to scale out. Amazon EC2 Auto Scaling does not scale your group below the minimum capacity or above the maximum capacity. To update your group, choose **Edit**, specify the minimum and maximum size, and then choose **Save**.

4. On the **Automatic scaling** tab, in **Scaling policies**, choose **Add policy**. (Old console: The **Scaling Policies** tab is where you can add the policy.)

5. To define a policy for scale out (increase capacity), do the following:

    a. For **Policy type**, choose **Step scaling**. (Old console: Choose **Create a scaling policy with steps**.)

    b. Specify a name for the policy.

    c. For **Execute policy when**, choose your alarm. If you haven't already created an alarm, do so now. Set the alarm threshold to greater than or equal to 80 percent.

    d.   Specify the change in the current group size that this policy will make when executed using **Take the action**. You can add a specific number of instances or a percentage of the existing group size, or set the group to an exact size.

          For example, choose `Add`, enter `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 positive infinity.

    e.   To add another step, choose **Add step** and then define the amount by which to scale and the lower and upper bounds of the step relative to the alarm threshold.

    f.   To set a minimum number of instances to scale, update the number field in **Add capacity units in increments of at least** `1` **capacity units**.

    g.   Specify an instance warm-up value for **Instances need**, which allows you to control the amount of time until a newly launched instance can contribute to the CloudWatch metrics.

6.   Choose **Create**.

To create a step scaling policy for scale in

1.   Choose **Add policy** to continue where you left off after creating a policy for scale out.

2.   To define a policy for scale in (decrease capacity), do the following:

    a.   For **Policy type**, choose **Step scaling**. (Old console: Choose **Create a scaling policy with steps**.)

    b.   Specify a name for the policy.

    c.   For **Execute policy when**, choose your alarm. If you haven't already created an alarm, do so now. Set the alarm threshold to less than or equal to 40 percent.

    d.   Specify the change in the current group size that this policy will make when executed using **Take the action**. You can remove a specific number of instances or a percentage of the existing group size, or set the group to an exact size.

          For example, choose `Remove`, enter `2` in the next field, and then choose `capacity units`. By default, the upper bound for this step adjustment is the alarm threshold and the lower bound is negative infinity.

    e.   To add another step, choose **Add step** and then define the amount by which to scale and the lower and upper bounds of the step relative to the alarm threshold.

3.   Choose **Create**.

Configure Step Scaling Policies (AWS CLI)

Use the AWS CLI as follows to configure step 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-launch-config`. 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-launch-config \
--vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
--max-size 5 --min-size 1
```

Step 2: Create Scaling Policies

You can create step or simple scaling policies that tell the Auto Scaling group what to do when the load on the application changes.

Example: my-step-scale-out-policy

Use the following **put-scaling-policy** command to create a step scaling policy named `my-step-scale-out-policy`, with an adjustment type of `PercentChangeInCapacity` that increases the capacity of the group based on the following step adjustments (assuming a CloudWatch alarm threshold of 60 percent):

- Increase the instance count by 10 percent when the value of the metric is greater than or equal to 70 percent but less than 80 percent
- Increase the instance count by 20 percent when the value of the metric is greater than or equal to 80 percent but less than 90 percent
- Increase the instance count by 30 percent when the value of the metric is greater than or equal to 90 percent

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg   \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
                    MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

The output includes the ARN for the policy. Store this ARN in a safe place. You need it to create CloudWatch alarms.

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy
}
```

Example: my-step-scale-in-policy

Use the following **put-scaling-policy** command to create a step scaling policy named `my-step-scale-in-policy`, with an adjustment type of `ChangeInCapacity` that decreases the capacity of the group by 2 instances.

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg   \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

The output includes the ARN that serves as a unique name for the policy. Later, 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 need it to create CloudWatch alarms.

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-
cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy
}
```

Examples of Simple Scaling Policies

Alternatively, you can create simple scaling policies by using the following CLI commands instead of the preceding CLI commands. The output of these commands includes an ARN for each policy, which you need to create the CloudWatch alarms. Keep in mind that a cooldown period will be in place due to the use of simple scaling policies.

Example: my-simple-scale-out-policy

Use the following **put-scaling-policy** command to create a simple scaling policy named `my-simple-scale-out-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-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity
```

Example: my-simple-scale-in-policy

Use the following **put-scaling-policy** command to create a simple scaling policy named `my-simple-scale-in-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-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -2 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

Step 3: Create CloudWatch Alarms

In step 2, you created scaling policies that provided instructions to the Auto Scaling group about how to scale in and scale out when the conditions that you specify change. In this 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 based on an average CPU threshold value of 60 percent for at least two consecutive evaluation periods of two minutes. To use your own custom metric, specify its name in `--metric-name` and its namespace in `--namespace`.

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 60 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --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 based on average CPU threshold value of 40 percent for at least two consecutive evaluation periods of two minutes. To use your own custom metric, specify its name in `--metric-name` and its namespace in `--namespace`.

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

Scaling Based on Amazon SQS

This section shows you how to scale your Auto Scaling group in response to changes in system load in an Amazon Simple Queue Service (Amazon SQS) queue. To learn more about how you can use Amazon SQS, see the Amazon Simple Queue Service Developer Guide.

There are some scenarios where you might think about scaling in response to activity in an Amazon SQS queue. For example, suppose that you have a web app that lets users upload images and use them online. In this scenario, each image requires resizing and encoding before it can be published. The app runs on EC2 instances in an Auto Scaling group, and it's configured to handle your typical upload rates. Unhealthy instances are terminated and replaced to maintain current instance levels at all times. The app places the raw bitmap data of the images in an SQS queue for processing. It processes the images and then publishes the processed images where they can be viewed by users. The architecture for this scenario works well if the number of image uploads doesn't vary over time. But if the number of uploads changes over time, you might consider using dynamic scaling to scale the capacity of your Auto Scaling group.

Using Target Tracking with the Right Metric

If you use a target tracking scaling policy based on a custom Amazon SQS queue metric, dynamic scaling can adjust to the demand curve of your application more effectively. For more information about choosing metrics for target tracking, see Choosing Metrics (p. 112).

The issue with using a CloudWatch Amazon SQS metric like `ApproximateNumberOfMessagesVisible` for target tracking is that the number of messages in the queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. That's because the number of messages in your SQS queue does not solely define the number of instances needed. The number of instances in your Auto Scaling group can be driven by multiple factors, including how long it takes to process a message and the acceptable amount of latency (queue delay).

The solution is to use a *backlog per instance* metric with the target value being the *acceptable backlog per instance* to maintain. You can calculate these numbers as follows:

- **Backlog per instance**: To calculate your backlog per instance, start with the `ApproximateNumberOfMessages` queue attribute to determine the length of the SQS queue (number of messages available for retrieval from the queue). Divide that number by the fleet's running capacity, which for an Auto Scaling group is the number of instances in the `InService` state, to get the backlog per instance.
- **Acceptable backlog per instance**: To calculate your target value, first determine what your application can accept in terms of latency. Then, take the acceptable latency value and divide it by the average time that an EC2 instance takes to process a message.

To illustrate with an example, let's say that the current `ApproximateNumberOfMessages` is 1500 and the fleet's running capacity is 10. If the average processing time is 0.1 seconds for each message and the longest acceptable latency is 10 seconds, then the acceptable backlog per instance is 10 / 0.1,
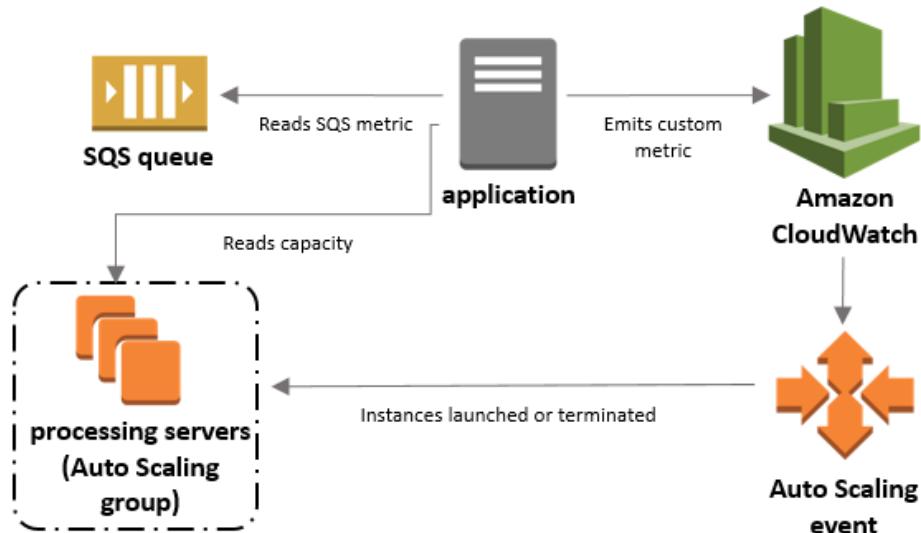
which equals 100. This means that 100 is the target value for your target tracking policy. If the backlog per instance is currently at 150 (1500 / 10), your fleet scales out, and it scale out by five instances to maintain proportion to the target value.

The following procedures demonstrate how to publish the custom metric and create the target tracking scaling policy that configures your Auto Scaling group to scale based on these calculations.

There are three main parts to this configuration:

- An Auto Scaling group to manage EC2 instances for the purposes of processing messages from an SQS queue.
- A custom metric to send to Amazon CloudWatch that measures the number of messages in the queue per EC2 instance in the Auto Scaling group.
- A target tracking policy that configures your Auto Scaling group to scale based on the custom metric and a set target value. CloudWatch alarms invoke the scaling policy.

The following diagram illustrates the architecture of this configuration.

Limitations and Prerequisites

To use this configuration, you need to be aware of the following limitations:

- You must use the AWS CLI or AWS SDKs to publish your custom metric to CloudWatch. You can then monitor your metric with the AWS Management Console.
- The SQS queue metric `ApproximateNumberOfMessages` is not in CloudWatch. You can only get the value of this metric using the SQS **get-queue-attributes** command or the GetQueueAttributes operation.
- After publishing your custom metric, you must use the AWS CLI or AWS SDKs to create a target tracking scaling policy with a customized metric specification.

The following sections direct you to use the AWS CLI for the tasks you need to perform. Make sure that you have the CLI installed and configured.

Before you begin, you must have an Amazon SQS queue to use. The following sections assume that you already have a queue (standard or FIFO), an Auto Scaling group, and EC2 instances running the application that uses the queue. For more information about Amazon SQS, see the Amazon Simple Queue Service Developer Guide.

Configure Scaling Based on Amazon SQS

Tasks

Step 1: Create a CloudWatch Custom Metric

A custom metric is defined using a metric name and namespace of your choosing. Namespaces for custom metrics cannot start with "AWS/". For more information about publishing custom metrics, see the Publish Custom Metrics topic in the *Amazon CloudWatch User Guide*.

Follow this procedure to create the custom metric by first reading information from your AWS account. Then, calculate the backlog per instance metric, as recommended in an earlier section. Lastly, publish this number to CloudWatch at a 1-minute granularity. Whenever possible, we strongly recommend that you scale on metrics with a 1-minute granularity to ensure a faster response to changes in system load.

To create a CloudWatch custom metric

1. Use the SQS **get-queue-attributes** command to get the number of messages waiting in the queue (`ApproximateNumberOfMessages`).

   ```
   aws sqs get-queue-attributes --queue-url https://sqs.region.amazonaws.com/123456789/MyQueue \
     --attribute-names ApproximateNumberOfMessages
   ```

2. Use the **describe-auto-scaling-groups** command to get the running capacity of the group, which is the number of instances in the `InService` lifecycle state. This command returns the instances of an Auto Scaling group along with their lifecycle state.

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

3. Calculate the backlog per instance by dividing the approximate number of messages available for retrieval from the queue by the fleet's running capacity.

4. Publish the results at a 1-minute granularity as a CloudWatch custom metric.

   Here is an example CLI **put-metric-data** command.

   ```
   aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --namespace MyNamespace \
     --unit None --value 20 --dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
   ```

After your application is emitting the desired metric, the data is sent to CloudWatch. The metric is visible in the CloudWatch console. You can access it by logging into the AWS Management Console and navigating to the CloudWatch page. Then, view the metric by navigating to the metrics page or by searching for it using the search box. For information about viewing metrics, see View Available Metrics in the *Amazon CloudWatch User Guide*.

Step 2: Create a Target Tracking Scaling Policy

After publishing your custom metric, create a target tracking scaling policy with a customized metric specification.

To create a target tracking scaling policy

1. Use the following command to specify a target value for your scaling policy in a `config.json` file in your home directory.

   For the `TargetValue`, calculate the acceptable backlog per instance metric and enter it here. To calculate this number, decide on a normal latency value and divide it by the average time that it takes to process a message.

   ```
   $ cat ~/config.json
   {
       "TargetValue":100,
       "CustomizedMetricSpecification":{
           "MetricName":"MyBacklogPerInstance",
           "Namespace":"MyNamespace",
           "Dimensions":[
               {
                   "Name":"MyOptionalMetricDimensionName",
                   "Value":"MyOptionalMetricDimensionValue"
               }
           ],
           "Statistic":"Average",
           "Unit":"None"
       }
   }
   ```

2. Use the **put-scaling-policy** command, along with the `config.json` file that you created in the previous step, to create your scaling policy.

   ```
   aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy
    \
      --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
      --target-tracking-configuration file://~/config.json
   ```

   This creates two alarms: one for scaling out and one for scaling in. It also returns the Amazon Resource Name (ARN) of the policy that is registered with CloudWatch, which CloudWatch uses to invoke scaling whenever the metric is in breach.

Step 3: Test Your Scaling Policy

After your setup is complete, verify that your scaling policy is working. You can test it by increasing the number of messages in your SQS queue and then verifying that your Auto Scaling group has launched an additional EC2 instance. You can also test it 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 function

1. Follow the steps in Tutorial: Sending a Message to an Amazon SQS Queue to add messages to your queue. Make sure that you have increased the number of messages in the queue so that the backlog per instance metric exceeds the target value.

   It can take a few minutes for your changes to trigger the CloudWatch alarm.

2. Use the **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 function

1. Follow the steps in Tutorial: Sending a Message to an Amazon SQS Queue to remove messages from the queue. Make sure that you have decreased the number of messages in the queue so that the backlog per instance metric is below the target value.

   It can take a few minutes for your changes to trigger the CloudWatch alarm.

2. Use the **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
```

Verifying a Scaling Activity for an Auto Scaling Group

After you create a scaling policy, Amazon EC2 Auto Scaling starts evaluating the policy against the metric. The metric alarm goes to ALARM state when the metric breaches the threshold for a specified number of evaluation periods. This means that a scaling policy could trigger a scaling action soon after it's created. After Amazon EC2 Auto Scaling changes capacity in response to a scaling policy, you can verify the scaling activity in your account. If you want to receive email notification from Amazon EC2 Auto Scaling informing you that a scaling action was triggered, follow the instructions in Getting Amazon SNS Notifications When Your Auto Scaling Group Scales (p. 162).

To view the scaling activities for your Auto Scaling group (new 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. Search for the name of the Auto Scaling group.

   The **Instances** column shows the number of instances currently running. While an instance is being launched or terminated, the **Status** column displays a status of "Updating capacity." Wait for a few minutes, and then refresh the view to see the latest status. After a scaling activity completes, notice that the **Instances** and **Desired capacity** columns show new values.

   > **Note**
   > If you're using instance weighting, the **Weighted capacity** column measures the number of capacity units that the group contains. If this column is hidden, choose the gear-shaped icon in the top-right corner of the section and then enable **Weighted capacity**.

4. Select the check box next to the Auto Scaling group.

   a. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances.

   b. On the **Instance management** tab, under **Instances**, you can view the status of the instances that are currently running. The **Lifecycle** column contains the state of your instances. Note that it takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to `InService`.

To view the scaling activities for your Auto Scaling group (AWS CLI)

Use the following **describe-scaling-activities** command.

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

The following is example output.

Scaling activities are ordered by start time. Activities still in progress are described first.

```
{
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-
b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-tracking-
policy changing the desired capacity from 3 to 2.  At 2020-02-11T18:34:31Z an instance
 was taken out of service in response to a difference between desired and actual capacity,
 shrinking the capacity from 3 to 2.  At 2020-02-11T18:34:31Z instance i-06c4794c2499af1df
 was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}"
    },
...
  ]
}
```

To verify the size of your Auto Scaling group (AWS CLI)

Use the [describe-auto-scaling-groups](#) command.

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

The following is example output, with details about the group and the currently running instances.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupARN": "arn",
            "ServiceLinkedRoleARN": "arn",
            "TargetGroupARNs": [],
            "SuspendedProcesses": [],
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "Tags": [],
            "EnabledMetrics": [],
            "LoadBalancerNames": [],
            "AutoScalingGroupName": "my-asg",
            "DefaultCooldown": 300,
            "MinSize": 1,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
```

```
                "ProtectedFromScaleIn": false,
                "AvailabilityZone": "us-west-2a",
                "LaunchTemplate": {
                    "LaunchTemplateName": "my-launch-template",
                    "Version": "1",
                    "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                },
                "InstanceId": "i-0c20ac468fa3049e8",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService"
            }
        ],
        "MaxSize": 5,
        "VPCZoneIdentifier": "subnet-c87f2be0",
        "HealthCheckGracePeriod": 300,
        "TerminationPolicies": [
            "Default"
        ],
        "CreatedTime": "2019-03-18T23:30:42.611Z",
        "AvailabilityZones": [
            "us-west-2a"
        ],
        "HealthCheckType": "EC2",
        "NewInstancesProtectedFromScaleIn": false,
        "DesiredCapacity": 2
    }
  ]
}
```

Disabling a Scaling Policy for an Auto Scaling Group

This topic describes how to temporarily disable a scaling policy so it won't initiate changes to the number of instances the Auto Scaling group contains. When you disable a scaling policy, the configuration details are preserved, so you can quickly re-enable the policy. This is easier than temporarily deleting a policy when you don't need it, and recreating it later.

When a scaling policy is disabled, the Auto Scaling group does not scale out or scale in for the metric alarms that are breached while the scaling policy is disabled. However, any scaling activities still in progress are not stopped.

Note that disabled scaling policies still count toward your quotas on the number of scaling policies that you can add to an Auto Scaling group.

To disable a scaling policy (new 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 **Automatic scaling** tab, in **Scaling policies**, select a scaling policy, and then choose **Actions**, **Disable**.

When you are ready to re-enable the scaling policy, repeat these steps and then choose **Actions**, **Enable**. After you re-enable a scaling policy, your Auto Scaling group may immediately initiate a scaling action if there are any alarms currently in ALARM state.

To disable a scaling policy (AWS CLI)

Use the **put-scaling-policy** command with the `--no-enabled` option as follows. Specify all options in the command as you would specify them when creating the policy.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
    --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
    --estimated-instance-warmup 360 \
    --target-tracking-configuration '{ "TargetValue": 70, "PredefinedMetricSpecification":
 { "PredefinedMetricType": "ASGAverageCPUUtilization" } }' \
    --no-enabled
```

To re-enable a scaling policy (AWS CLI)

Use the **put-scaling-policy** command with the `--enabled` option as follows. Specify all options in the command as you would specify them when creating the policy.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
    --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
    --estimated-instance-warmup 360 \
    --target-tracking-configuration '{ "TargetValue": 70, "PredefinedMetricSpecification":
 { "PredefinedMetricType": "ASGAverageCPUUtilization" } }' \
    --enabled
```

To describe a scaling policy (AWS CLI)

Use the **describe-policies** command to verify the enabled status of a scaling policy.

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

The following is example output.

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "my-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 70.0,
                "DisableScaleIn": false
            },
            "Enabled": true
```

```
        }
    ]
}
```

Deleting a Scaling Policy

After you no longer need a scaling policy, you can delete it. Depending on the type of scaling policy, you might also need to delete the CloudWatch alarms. Deleting a target tracking scaling policy also deletes any associated CloudWatch alarms. Deleting a step scaling policy or a simple scaling policy deletes the underlying alarm action, but it does not delete the CloudWatch alarm, even if it no longer has an associated action.

To delete a scaling policy (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 **Automatic scaling** tab, in **Scaling policies**, select a scaling policy, and then choose **Actions**, **Delete**. (Old console: The **Scaling Policies** tab is where you can delete the policy.)
5. When prompted for confirmation, choose **Yes, Delete**.
6. (Optional) If you deleted a step scaling policy or a simple scaling policy, do the following to delete the CloudWatch alarm that was associated with the policy. You can skip these substeps to keep the alarm for future use.

    a. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
    b. On the navigation pane, choose **Alarms**.
    c. Choose the alarm (for example, `Step-Scaling-AlarmHigh-AddCapacity`) and choose **Action**, **Delete**.
    d. When prompted for confirmation, choose **Delete**.

To delete your scaling policy (AWS CLI)

Use the following **delete-policy** command.

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \
  --policy-name my-step-scale-in-policy
```

To delete your CloudWatch alarm (AWS CLI)

For step scaling policies and simple scaling policies, use the **delete-alarms** command to delete the CloudWatch alarm that was associated with the policy. You can skip this step to keep the alarm for future use. You can delete one or more alarms at a time. For example, use the following command to delete the `Step-Scaling-AlarmHigh-AddCapacity` and `Step-Scaling-AlarmLow-RemoveCapacity` alarms.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-
AlarmLow-RemoveCapacity
```

Example Scaling Policies for the AWS Command Line Interface (AWS CLI)

You can create scaling policies for Amazon EC2 Auto Scaling through the AWS Management Console, AWS CLI, or AWS SDKs.

The following examples show how you can create scaling policies with the AWS CLI **put-scaling-policy** command. For introductory exercises for creating scaling policies from the AWS CLI, see Target Tracking Scaling Policies (p. 110) and Step and Simple Scaling Policies (p. 114).

Example 1: To apply a target tracking scaling policy with a predefined metric specification

```
aws autoscaling put-scaling-policy --policy-name cpu40-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ASGAverageCPUUtilization"
  }
}
```

Example 2: To apply a target tracking scaling policy with a customized metric specification

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

Example 3: To apply a target tracking scaling policy for scale out only

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/EC2Co-EcsEl-1TKLTMITMM0EO/f37c06a68c1748aa/targetgroup/EC2Co-
Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
  },
  "DisableScaleIn": true
}
```

Example 4: To apply a step scaling policy for scale out

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \
```

```
 MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
                    MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
   --min-adjustment-magnitude 1
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 5: To apply a step scaling policy for scale in

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 6: To apply a simple scaling policy for scale out

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Example 7: To apply a simple scaling policy for scale in

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -2 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

Record the policy's Amazon Resource Name (ARN). You need the ARN when you create the CloudWatch alarm.

Scaling Cooldowns for Amazon EC2 Auto Scaling

The cooldown period helps you prevent your Auto Scaling group from launching or terminating additional instances before the effects of previous activities are visible. You can configure the length of time based on your instance start-up time or other application needs.

When you use a simple scaling policy for dynamic scaling, after the Auto Scaling group scales using a simple scaling policy, it waits for the cooldown period to complete before any further scaling activities due to simple scaling policies can start. An adequate cooldown period helps to prevent the initiation of a scaling activity based on stale metrics. By default, all simple scaling policies use the default cooldown period associated with your Auto Scaling group, but you can configure a different cooldown period for certain policies, as described in the following sections.

Note that Amazon EC2 Auto Scaling honors cooldown periods when using simple scaling policies, but not when using other scaling policies or scheduled scaling. For more information about simple scaling, see Step and Simple Scaling Policies (p. 114).

Important
In most cases, a target tracking scaling policy or a step scaling policy is more optimal for scaling performance than waiting for a fixed period of time to pass after there is a scaling activity. For a scaling policy that changes the size of your Auto Scaling group proportionally as the value of the scaling metric decreases or increases, we recommend target tracking over either simple scaling or step scaling.
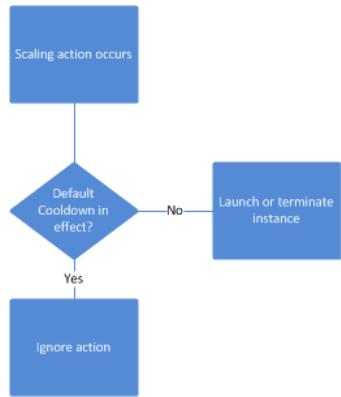
When you manually scale your Auto Scaling group using the AWS CLI or AWS SDKs, the default is not to wait for the cooldown period to complete, but you can override this behavior and honor the cooldown period. If an instance becomes unhealthy, the Auto Scaling group does not wait for the cooldown period to complete before replacing the unhealthy instance. For more information about manual scaling, see Manual Scaling for Amazon EC2 Auto Scaling (p. 96).

Contents

Default Cooldown Period

A default cooldown period automatically applies to any scaling activities for simple scaling policies, and you can optionally request to have it apply to your manual scaling activities.

When you use the AWS Management Console to update an Auto Scaling group, or when you use the AWS CLI or an AWS SDK to create or update an Auto Scaling group, you can set the optional default cooldown parameter. If a value for the default cooldown period is not provided, its default value is 300 seconds.

To modify a default cooldown period (console)

Create the Auto Scaling group in the usual way. After creating the Auto Scaling group, edit the group to specify the default cooldown period.

To modify a default cooldown period (AWS CLI)

Use one of the following commands:

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

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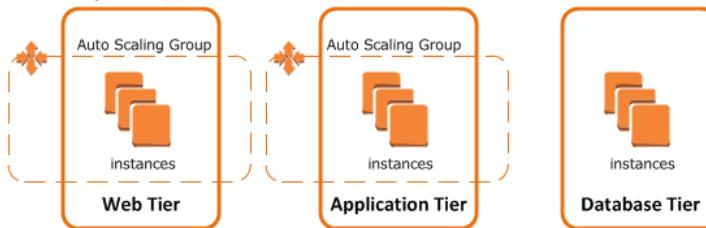
In addition to specifying the default cooldown period for your Auto Scaling group, you can create cooldowns that apply to a specific simple scaling policy and manual scaling. A scaling-specific cooldown period overrides the default cooldown period.

One common use for a scaling-specific cooldown period is with a scale-in policy. Because this policy terminates instances, Amazon EC2 Auto Scaling needs less time to determine whether to terminate additional instances. Terminating instances should be a much quicker operation than launching instances. The default cooldown period of 300 seconds is therefore too long. In this case, a scaling-specific cooldown period with a lower value of 180 seconds for your scale-in policy can help you reduce costs by allowing the group to scale in faster.

To specify a scaling-specific cooldown period, use the optional cooldown parameter when you create or update a simple scaling policy. For more information, see Step and Simple Scaling Policies (p. 114).

Example Simple Scaling Cooldown Scenario
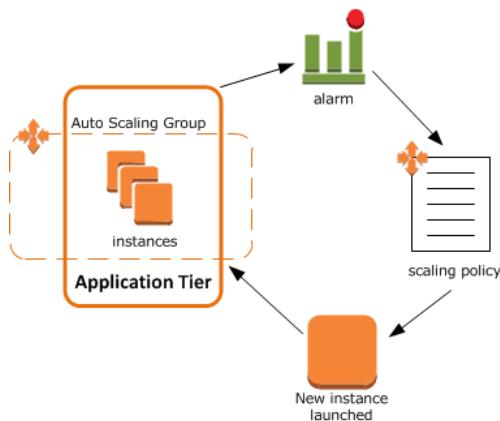
Consider the following scenario: you have a web application running in AWS. This web application consists of three basic tiers: web, application, and database. To make sure that the application always has the resources to meet traffic demands, you create two Auto Scaling groups: one for your web tier and one for your application tier.

To help ensure that the group for the application tier has the appropriate number of EC2 instances, you create a simple scaling policy to scale out whenever the value for the CloudWatch metric associated with the scaling policy exceeds a specified threshold for consecutive specified periods. When the CloudWatch alarm triggers the scaling policy, the Auto Scaling group 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's fully in service. The actual time 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, the Auto Scaling group 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 every minute for any standard resolution alarm, causing the Auto Scaling group to launch another instance each time the alarm fires.

However, with a cooldown period in place, the Auto Scaling group launches an instance and then blocks scaling activities due to simple scaling policies 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 scaling activities that are triggered after the cooldown period can resume. If the CloudWatch alarm fires again, the Auto Scaling group launches another instance, and the cooldown period takes effect again. However, if the additional instance was enough to bring the metric value back down, the group remains at its current size.

Cooldowns and Multiple Instances

The preceding section provides examples that show how cooldown periods affect Auto Scaling groups when a single instance launches or terminates. However, it is common for Auto Scaling groups to launch more than one instance at a time. For example, you might choose to have the Auto Scaling group 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 finishes launching or terminating.

Cooldowns and Lifecycle Hooks

You have the option to add lifecycle hooks to your Auto Scaling groups. These hooks enable you to control how instances launch and terminate within an Auto Scaling group so that you can perform custom actions on an instance before it is put into service or before it is terminated. When a lifecycle action occurs, and an instance enters the wait state, scaling activities due to simple scaling policies are paused. For more information, see Amazon EC2 Auto Scaling Lifecycle Hooks (p. 142).

Lifecycle hooks can affect the start time of any cooldown periods configured for the Auto Scaling group. For example, consider an Auto Scaling group that has a lifecycle hook that supports a custom action at instance launch. When the application experiences an increase in demand due to a simple scaling policy, the group launches an instance 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 activities due to simple scaling policies are paused. When the instance enters the `InService` state, the cooldown period starts. When the cooldown period expires, any scaling activities that are triggered after the cooldown period can resume.

Not all cooldowns are applied after the execution of lifecycle hooks

Usually, when an instance is terminating, the cooldown period does not begin until after the instance moves out of the `Terminating:Wait` state (after the lifecycle hook execution is complete).

However, with Elastic Load Balancing, the Auto Scaling group starts the cooldown period when the terminating instance finishes connection draining (deregistration delay) by the load balancer and does

not wait for the lifecycle hook. This is helpful for groups that have simple scaling policies for both scale in and scale out. The intention of a cooldown period is to allow the next scaling activity to occur as soon as the effects of the previous activities are visible. If an instance is terminating and then demand for your application suddenly increases, any scaling activities due to simple scaling policies that are paused can resume after connection draining and a cooldown period finishes. Otherwise, waiting to complete all three activities—connection draining, a lifecycle hook, and a cooldown period— significantly increases the amount of time that the Auto Scaling group needs to pause scaling.

Controlling Which Auto Scaling Instances Terminate During Scale In

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

This topic describes the default termination policy and the options available to you to configure your own customized termination policies. Using termination policies, you can control which instances you prefer to terminate first when a scale-in event occurs.

It also describes how to enable instance scale-in protection to prevent specific instances from being terminated during automatic scale in. For instances in an Auto Scaling group, use Amazon EC2 Auto Scaling features to protect an instance when a scale-in event occurs. If you want to protect your instance from being accidentally terminated, use Amazon EC2 termination protection.

> **Note**
> Auto Scaling groups with different types of purchase options (p. 48) are a unique situation. Amazon EC2 Auto Scaling first identifies which of the two types (Spot or On-Demand) should be terminated. If you balance your instances across Availability Zones, it chooses the Availability Zone with the most instances of that type to maintain balance. Then, it applies the default or customized termination policy.

Contents

Default Termination Policy

The default termination policy is designed to help ensure that your instances span Availability Zones evenly for high availability. The default policy is kept generic and flexible to cover a range of scenarios.

The default termination policy behavior is as follows:

1. Determine which Availability Zones have the most instances, and at least one instance that is not protected from scale in.
2. Determine which instances to terminate so as to align the remaining instances to the allocation strategy for the On-Demand or Spot Instance that is terminating. This only applies to an Auto Scaling group that specifies allocation strategies (p. 48).

   For example, after your instances launch, you change the priority order of your preferred instance types. When a scale-in event occurs, Amazon EC2 Auto Scaling tries to gradually shift the On-Demand Instances away from instance types that are lower priority.

3. Determine whether any of the instances use the oldest launch template or configuration:

   a. [For Auto Scaling groups that use a launch template]

      Determine whether any of the instances use the oldest launch template unless there are instances that use a launch configuration. Amazon EC2 Auto Scaling terminates instances that use a launch configuration before instances that use a launch template.

   b. [For Auto Scaling groups that use a launch configuration]
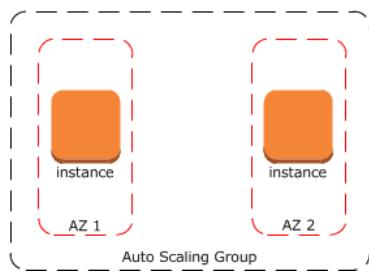
      Determine whether any of the instances use the oldest launch configuration.

4. After applying all of the above criteria, if there are multiple unprotected instances to terminate, determine which instances are closest to the next billing hour. If there are multiple unprotected instances closest to the next billing hour, terminate one of these instances at random.
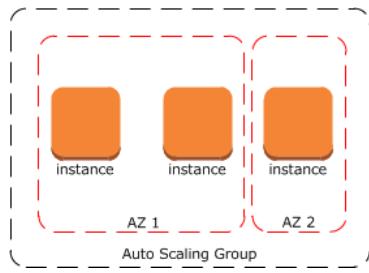
   Note that terminating the instance closest to the next billing hour helps you maximize the use of your instances that have an hourly charge. Alternatively, if your Auto Scaling group uses Amazon Linux or Ubuntu, your EC2 usage is billed in one-second increments. For more information, see Amazon EC2 Pricing.

Example

Consider an Auto Scaling group that uses a launch configuration. It has one instance type, two Availability Zones, a desired capacity of two instances, and scaling policies that increase and decrease the number of instances by one 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 the Auto Scaling group 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 the Auto Scaling group terminates one of the instances. If you did not assign a specific termination policy to the group, it uses the default termination policy. It 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, the Auto Scaling group selects the instance that is closest to the next billing hour and terminates it.

For more information about high availability with Amazon EC2 Auto Scaling, see Distributing Instances Across Availability Zones (p. 6).

Customizing the Termination Policy

You have the option of replacing the default policy with a customized one to support common use cases like keeping instances that have the current version of your application.

When you customize the termination policy, if one Availability Zone has more instances than the other Availability Zones that are used by the group, your termination policy is applied to the instances from the imbalanced Availability Zone. If the Availability Zones used by the group are balanced, the termination policy is applied across all of the Availability Zones for the group.

Amazon EC2 Auto Scaling supports the following custom termination policies:

- `OldestInstance`. Terminate 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. You can gradually replace instances of the old type with instances of the new type.
- `NewestInstance`. Terminate 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`. Terminate 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`. Terminate instances that are closest to the next billing hour. This policy helps you maximize the use of your instances that have an hourly charge.
- `Default`. Terminate instances according to the default termination policy. This policy is useful when you have more than one scaling policy for the group.
- `OldestLaunchTemplate`. Terminate instances that have the oldest launch template. With this policy, instances that use the noncurrent launch template are terminated first, followed by instances that use the oldest version of the current launch template. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
- `AllocationStrategy`. Terminate instances in the Auto Scaling group to align the remaining instances to the allocation strategy for the type of instance that is terminating (either a Spot Instance or an On-Demand Instance). This policy is useful when your preferred instance types have changed. If the Spot allocation strategy is `lowest-price`, you can gradually rebalance the distribution of Spot Instances across your N lowest priced Spot pools. If the Spot allocation strategy is `capacity-optimized`, you can gradually rebalance the distribution of Spot Instances across Spot pools where there is more available Spot capacity. You can also gradually replace On-Demand Instances of a lower priority type with On-Demand Instances of a higher priority type.

To customize a termination policy (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** in **Advanced configuration**. Choose one or more termination policies. If you choose multiple policies, list them in the order in which they should apply. If you use the **Default** policy, make it the last one in the list.
6. Choose **Save**.

To customize a termination policy (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. For example, use the following command to update an Auto Scaling group to use the `OldestLaunchConfiguration` policy first and then 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 Scale-In Protection

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

Instance scale-in protection starts when the instance state is `InService`. If you detach an instance that is protected from termination, its instance scale-in protection setting is lost. When you attach the instance to the group again, it inherits the current instance scale-in 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, its desired capacity is decremented. However, the Auto Scaling group can't terminate the required number of instances until their instance scale-in protection settings are disabled.

Instance scale-in protection does not protect Auto Scaling instances from the following:

- Manual termination through the Amazon EC2 console, the `terminate-instances` command, or the `TerminateInstances` action. To protect Auto Scaling instances from manual termination, enable Amazon EC2 termination protection. For more information, see Enabling Termination Protection in the *Amazon EC2 User Guide for Linux Instances*.
- Health check replacement if the instance fails health checks. For more information, see Health Checks for Auto Scaling Instances (p. 160). To prevent Amazon EC2 Auto Scaling from terminating unhealthy instances, suspend the `ReplaceUnhealthy` process. For more information, see Suspending and Resuming Scaling Processes (p. 154).
- Spot Instance interruptions. A Spot Instance is terminated when capacity is no longer available or the Spot price exceeds your maximum price.

Tasks

Enable Instance Scale-In Protection for a Group

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

To enable instance scale-in protection (new console)

When you create the Auto Scaling group, on the **Configure group size and scaling policies** page, under **Instance scale-in protection**, select the **Enable instance scale-in protection** option.

To enable instance scale-in protection (old 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 scale-in protection (AWS CLI)

Use the following **create-auto-scaling-group** command to enable instance scale-in protection.

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

Modify the Instance Scale-In Protection Setting for a Group

You can enable or disable the instance scale-in protection setting for an Auto Scaling group. When the
instance scale-in protection setting is enabled, all new instances launched after enabling it will have
instance scale-in protection enabled. Previously launched instances are not protected from scale in
unless you enable the instance scale-in protection setting for each instance individually.

To change the instance scale-in protection setting for a group (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. In **Advanced configuration**, select **Enable instance scale-in protection on all new instances by
   default** from **Instance scale-in protection**. (Old console: For **Instance Protection**, select **Protect
   From Scale In**.)
6. Choose **Save**.

To change the instance scale-in protection setting for a group (AWS CLI)

Use the following **update-auto-scaling-group** command to enable instance scale-in 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 scale-in 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 Scale-In Protection Setting for an Instance

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

To change the instance scale-in protection setting for an instance (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 **Instance management** tab, in **Instances**, select an instance. (Old console: The **Instances** tab
   is where you can select the instance.)

5.   To enable instance scale-in protection, choose **Actions**, **Set scale-in protection**. When prompted, choose **Set scale-in protection**.

6.   To disable instance scale-in protection, choose **Actions**, **Remove scale-in protection**. When prompted, choose **Remove scale-in protection**.

To change the instance scale-in protection setting for an instance (AWS CLI)

Use the following **set-instance-protection** command to enable instance scale-in 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 scale-in 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
```

Amazon EC2 Auto Scaling Lifecycle Hooks

Lifecycle hooks enable you to perform custom actions by *pausing* instances as an Auto Scaling group launches or terminates them. When an instance is paused, it remains in a wait state either until you complete the lifecycle action using the **complete-lifecycle-action** command or the `CompleteLifecycleAction` operation, or until the timeout period ends (one hour by default).

For example, let's say that your newly launched instance completes its startup sequence and a lifecycle hook pauses the instance. While the instance is in a wait state, you can install or configure software on it, making sure that your instance is fully ready before it starts receiving traffic. For another example of the use of lifecycle hooks, let's say that when a scale-in event occurs, the terminating instance is first deregistered from the load balancer (if the Auto Scaling group is being used with Elastic Load Balancing). Then, a lifecycle hook pauses the instance before it is terminated. While the instance is in the wait state, you can, for example, connect to the instance and download logs or other data before the instance is fully terminated.

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 Amazon EC2 Auto Scaling Service Quotas (p. 9).

Contents

How Lifecycle Hooks Work

After you add lifecycle hooks to your Auto Scaling group, they work as follows:

1. The Auto Scaling group responds to scale-out events by launching instances and scale-in events by terminating instances.

2. The lifecycle hook puts the instance into a wait state (`Pending:Wait` or `Terminating:Wait`). The instance is paused until you 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 Amazon EC2 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. Amazon EC2 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 the Auto Scaling group 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

Adding lifecycle hooks to your Auto Scaling group gives you greater control over how instances launch and terminate. The following are things to consider when adding a lifecycle hook to your Auto Scaling group, to help ensure that the group continues to perform as expected.

Keeping Instances in a Wait State

Instances can remain in a wait state for a finite period of time. The default is one 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.
- Postpone the end of the timeout period by recording a heartbeat, using the **record-lifecycle-action-heartbeat** command or the `RecordLifecycleActionHeartbeat` operation. This extends the timeout period by the timeout value specified when you created the lifecycle hook. For example, if the timeout value is one 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.

Cooldown Periods for Simple Scaling

When an Auto Scaling group launches or terminates an instance due to a simple scaling policy, a cooldown period takes effect. The cooldown period helps ensure that the Auto Scaling group does not launch or terminate more instances than needed before the effects of previous simple scaling activities are visible. When a lifecycle action occurs, and an instance enters the wait state, scaling activities due to simple scaling policies are paused. When the lifecycle hook execution finishes, the cooldown period starts. If you set a long interval for the cooldown period, it will take more time for scaling to resume. For more information, see Scaling Cooldowns for Amazon EC2 Auto Scaling (p. 133).

Health Check Grace Period

If you add a lifecycle hook, the health check grace period does not start until the lifecycle hook actions complete and the instance enters the `InService` state.

Spot Instances

You can use lifecycle hooks with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating in the event that capacity is no longer available. 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 configure notifications for when an instance enters a wait state. You can use Amazon CloudWatch Events, Amazon SNS, or Amazon SQS to receive the notifications. For more information, see Configuring Lifecycle Hook Notifications (p. 147).

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 in the *Amazon EC2 User Guide for Linux Instances*.

Add Lifecycle Hooks

When you add a lifecycle hook to your Auto Scaling group, you can specify whether it should be run when instances launch or terminate in the Auto Scaling group.

Contents
- Add Lifecycle Hooks (Console) (p. 145)
- Add Lifecycle Hooks (AWS CLI} (p. 145)

Add Lifecycle Hooks (Console)

Follow these steps to add a lifecycle hook to an existing Auto Scaling group. You can specify whether the hook is used when the instances launch or terminate and how long to wait until the lifecycle hook is completed before abandoning or continuing.

To add a lifecycle hook

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 **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**. (Old console: The **Lifecycle Hooks** tab is where you can create a lifecycle hook.)
5. To define a lifecycle hook, do the following:

    a. For **Lifecycle hook name**, specify a name for the lifecycle hook.

    b. For **Lifecycle transition**, choose **Instance launch** or **Instance terminate**.

    c. Specify a timeout value for **Heartbeat timeout**, which allows you to control the amount of time for the instances to remain in a wait state. The value must be from 30 to 7200 seconds. During the timeout period, you can, for example, log on to a newly launched instance, and install applications or perform custom actions.

    d. For **Default result**, specify the action that the Auto Scaling group takes when the lifecycle hook timeout elapses or if an unexpected failure occurs. You can choose to either ABANDON or CONTINUE.

    If the instance is launching, CONTINUE indicates that your actions were successful, and that the Auto Scaling group 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, and CONTINUE allows any other lifecycle hooks to complete.

    e. (Optional) For **Notification metadata**, specify additional information that you want to include any time that Amazon EC2 Auto Scaling sends a message to the notification target.

6. Choose **Create**.

Add Lifecycle Hooks (AWS CLI}

Create and update lifecycle hooks using the **put-lifecycle-hook** command.

To perform an action on scale out, use the following command.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-hook --auto-scaling-group-
name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

To perform an action on scale in, use the following command instead.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-hook --auto-scaling-group-
name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

To receive notifications using Amazon SNS or Amazon SQS, you must specify a notification target and an IAM role. For more information, see Configuring Notifications for Amazon EC2 Auto Scaling Lifecycle Hooks (p. 147).

For example, add the following options to specify an SNS topic as the notification target.

```
--notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic --role-arn
 arn:aws:iam::123456789012:role/my-notification-role
```

The topic receives a test notification with the following key-value pair.

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

Complete a Lifecycle Hook Custom Action

When an Auto Scaling group responds to a scale-out or scale-in event, it puts the instance in a wait state and, depending on how the lifecycle hook is configured, sends a notification.

To complete a lifecycle hook custom action

1. After receiving a notification, you can perform a custom action while the instance is in a wait state.

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 that you received with the notification, as shown in the following command.

    ```
    aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
      --auto-scaling-group-name my-asg --lifecycle-action-
    token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
    ```

    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 --lifecycle-hook-name my-launch-hook \
      --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
    ```

3. If you finish the custom action before the timeout period ends, use the **complete-lifecycle-action** command so that the Auto Scaling group can continue launching or terminating the instance. You can specify the lifecycle action token, as shown in the following command.

    ```
    aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
      --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \
      --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
    ```

    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. You receive a notification within a few minutes after instance launch.

To change the desired capacity (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 capacity**, increase the current value by 1. If this value exceeds **Maximum capacity**, you must also increase the value of **Maximum capacity** 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 capacity** by 1. After a few minutes, you'll receive notification for the event.

Configuring Notifications for Amazon EC2 Auto Scaling Lifecycle Hooks

You can add a lifecycle hook to an Auto Scaling group that triggers a notification when an instance enters a wait state. You can configure these notifications for a variety of reasons, for example, to invoke a Lambda function or to receive email notification so that you can perform a custom action. This topic describes how to configure notifications using Amazon CloudWatch Events, Amazon SNS, and Amazon SQS. 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.

> **Important**
> AWS resources for notifications must always be created in the same AWS Region where you create your lifecycle hook. For example, if you configure notifications using Amazon SNS, the Amazon SNS topic must reside in the same Region as your lifecycle hook.

Notification Options

Route Notifications to Lambda 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. 171) and note its Amazon Resource Name (ARN). For example, `arn:aws:lambda:region: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 following examples shows the `pattern.json` for an instance launch lifecycle action.

```
{
```

```
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

The following examples shows the `pattern.json` for an instance terminate lifecycle action.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ]
}
```

3. Grant the rule permission to invoke your Lambda function using the following **add-permission** command. This command trusts the CloudWatch Events service principal (`events.amazonaws.com`) and scopes permissions to the specified rule.

```
aws lambda add-permission --function-name LogScheduledEvent --statement-id my-scheduled-event \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
 arn:aws:events:region:123456789012:rule/my-scheduled-rule
```

4. 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:region:123456789012:function:my-function
```

5. After you have followed these instructions, continue on to Add Lifecycle Hooks (p. 144) as a next step.

When the Auto Scaling group 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. 168).

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 an Amazon SNS topic using the following **create-topic** command. For more information, see Create a Topic in the *Amazon Simple Notification Service Developer Guide*.

```
aws sns create-topic --name my-sns-topic
```

Note the ARN of the target (for example, `arn:aws:sns:region:123456789012:my-sns-topic`).

2. Create a service role (or *assume* role) for Amazon EC2 Auto Scaling to which you can grant permission to access your notification target.

To create an IAM role and allow Amazon EC2 Auto Scaling to assume it

a. Open the IAM console at https://console.aws.amazon.com/iam/.

b. In the navigation pane, choose **Roles**, **Create new role**.

c. Under **Select type of trusted entity**, choose **AWS service**.

d. Under **Choose the service that will use this role**, choose **EC2 Auto Scaling** from the list.

e.   Under **Select your use case**, choose **EC2 Auto Scaling Notification Access**, and then choose **Next:Permissions**.

f.   Choose **Next:Tags**. Optionally, you can add metadata to the role by attaching tags as key–value pairs. Then choose **Next:Review**.

g.   On the **Review** page, enter a name for the role (for example, my-notification-role), and choose **Create role**.

h.   On the **Roles** page, choose the role that you just created to open the **Summary** page. Make a note of the **Role ARN**. For example, `arn:aws:iam::`*`123456789012`*`:role/`*`my-notification-role`*. You will specify the role ARN when you create the lifecycle hook in the next procedure.

3.   After you have followed these instructions, continue on to as a next step.

When the Auto Scaling group 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, a message is published 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.

The following is a notification message example.

```
Service: AWS Auto Scaling
Time: 2019-04-30T20:42:11.305Z
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.

Important
FIFO queues are not compatible with lifecycle hooks.

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 (for example, `arn:aws:sqs:`*`region`*`:`*`123456789012`*`:`*`my-sqs-queue`*).

2.   Create a service role (or *assume* role) for Amazon EC2 Auto Scaling to which you can grant permission to access your notification target.

   To create an IAM role and allow Amazon EC2 Auto Scaling to assume it

a. Open the IAM console at https://console.aws.amazon.com/iam/.

b. In the navigation pane, choose **Roles**, **Create new role**.

c. Under **Select type of trusted entity**, choose **AWS service**.

d. Under **Choose the service that will use this role**, choose **EC2 Auto Scaling** from the list.

e. Under **Select your use case**, choose **EC2 Auto Scaling Notification Access**, and then choose **Next:Permissions**.

f. Choose **Next:Tags**. Optionally, you can add metadata to the role by attaching tags as key–value pairs. Then choose **Next:Review**.

g. On the **Review** page, enter a name for the role (for example, my-notification-role), and choose **Create role**.

h. On the **Roles** page, choose the role you that just created to open the **Summary** page. Make a note of the **Role ARN**. For example, `arn:aws:iam::`*`123456789012`*`:role/`*`my-notification-role`*. You will specify the role ARN when you create the lifecycle hook in the next procedure.

3. After you have followed these instructions, continue on to Add Lifecycle Hooks (AWS CLI} (p. 145) as a next step.

When the Auto Scaling group 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, a message is published to the notification target.

Temporarily Removing Instances from Your Auto Scaling Group

You can 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 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, the value that you specified as your desired capacity is decremented when you put an instance on standby. This prevents the launch of an additional instance while you have this instance on standby. Alternatively, you can specify that your desired capacity is not decremented. If you specify this option, the Auto Scaling group launches an instance to replace the one on standby. The intention is to help you maintain capacity for your application while one or more instances are on standby.

4. You can update or troubleshoot the instance.

5. You return the instance to service by exiting the standby state.

6. After you put an instance that was on standby back in service, the desired capacity is incremented. If you did not decrement the capacity when you put the instance on standby, the Auto Scaling group detects that you have more instances than you need. It applies the termination policy in effect to reduce the size of the group. For more information, see Controlling Which Auto Scaling Instances Terminate During Scale In (p. 137).

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

Amazon EC2 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. Amazon EC2 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, Amazon EC2 Auto Scaling continues to report the instance as healthy. If you attempt to put a terminated instance that was on standby back in service, Amazon EC2 Auto Scaling performs a health check on the instance, determines that it is terminating and unhealthy, and launches a replacement instance.

Temporarily Remove an Instance (Console)

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance

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 **Instance management** tab, in **Instances**, select an instance. (Old console: The **Instances** tab is where you can select the instance.)

5. Choose **Actions**, **Set to Standby**.

6. In the **Set to Standby** dialog box, select the check box to launch a replacement instance. 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**. In the **Set to InService** dialog box, choose **Set to InService**.

Temporarily Remove an Instance (AWS CLI)

The following procedure demonstrates the general process for updating an instance that is currently in service.

To temporarily remove an instance

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": [
           {
               "ProtectedFromScaleIn": false,
               "AvailabilityZone": "us-west-2a",
               "LaunchTemplate": {
                   "LaunchTemplateName": "my-launch-template",
                   "Version": "1",
                   "LaunchTemplateId": "lt-050555ad16a3f9c7f"
               },
               "InstanceId": "i-05b4f7d5be44822a6",
               "AutoScalingGroupName": "my-asg",
               "HealthStatus": "HEALTHY",
               "LifecycleState": "InService"
           },
           ...
       ]
   }
   ```

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 the Auto Scaling group does not launch a replacement instance.

   ```
   aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
     --auto-scaling-group-name my-asg --should-decrement-desired-capacity
   ```

   The following is an example response.

   ```
   {
       "Activities": [
           {
               "Description": "Moving EC2 instance to Standby: i-05b4f7d5be44822a6",
               "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-05b4f7d5be44822a6 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-05b4f7d5be44822a6
```

The following is an example response. Notice that the status of the instance is now `Standby`.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby"
        },
        ...
    ]
}
```

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-05b4f7d5be44822a6 --auto-scaling-group-
name my-asg
```

The following is an example response.

```
{
    "Activities": [
        {
            "Description": "Moving EC2 instance out of Standby: i-05b4f7d5be44822a6",
            "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-05b4f7d5be44822a6 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-05b4f7d5be44822a6
```

The following is an example response. Notice that the status of the instance is `InService`.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        ...
    ]
}
```

Suspending and Resuming Scaling Processes

This topic explains how to suspend and then resume one or more of the scaling processes for your Auto Scaling group. It also describes the issues to consider when choosing to use the suspend-resume feature of Amazon EC2 Auto Scaling.

Important
Use the standby feature instead of the suspend-resume feature if you need to troubleshoot or reboot an instance. For more information, see Temporarily Removing Instances from Your Auto Scaling Group (p. 150). Use the instance scale-in protection feature to prevent specific instances from being terminated during automatic scale in. For more information, see Instance Scale-In Protection (p. 140).

In addition to suspensions that you initiate, Amazon EC2 Auto Scaling can also suspend processes for Auto Scaling groups that repeatedly fail to launch instances. This is known as an *administrative suspension*. An administrative suspension 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 that were suspended by Amazon EC2 Auto Scaling for administrative reasons.

Contents

Scaling Processes

For Amazon EC2 Auto Scaling, there are two primary process types: `Launch` and `Terminate`. The `Launch` process adds a new Amazon EC2 instance to an Auto Scaling group, increasing its capacity. The `Terminate` process removes an Amazon EC2 instance from the group, decreasing its capacity.

The other process types for Amazon EC2 Auto Scaling relate to specific scaling features:

- `AddToLoadBalancer`—Adds instances to the attached load balancer or target group when they are launched.
- `AlarmNotification`—Accepts notifications from CloudWatch alarms that are associated with the group's scaling policies.
- `AZRebalance`—Balances the number of EC2 instances in the group evenly across all of the specified Availability Zones when the group becomes unbalanced, for example, a previously unavailable Availability Zone returns to a healthy state. For more information, see Rebalancing Activities (p. 7).
- `HealthCheck`—Checks the health of the instances and marks an instance as unhealthy if Amazon EC2 or Elastic Load Balancing tells Amazon EC2 Auto Scaling that the instance is unhealthy. This process can override the health status of an instance that you set manually. For more information, see Health Checks for Auto Scaling Instances (p. 160).
- `ReplaceUnhealthy`—Terminates instances that are marked as unhealthy and then creates new instances to replace them.
- `ScheduledActions`—Performs the scheduled scaling actions that you create or that are created by the predictive scaling feature of AWS Auto Scaling.

Choosing to Suspend

Each process type can be suspended and resumed independently. This section provides some guidance and behavior to take into account before deciding to suspend a scaling process. Keep in mind that suspending individual processes might interfere with other processes. Depending on the reason for suspending a process, you might need to suspend multiple processes together.

The following descriptions explain what happens when individual process types are suspended.

> **Warning**
> If you suspend either the `Launch` or `Terminate` process types, it can prevent other process types from functioning properly.

`Terminate`

- Your Auto Scaling group does not scale in for alarms or scheduled actions that occur while the process is suspended. In addition, the following processes are disrupted:
  - `AZRebalance` is still active but does not function properly. It can launch new instances without terminating the old ones. This could cause your Auto Scaling group to grow up to 10 percent larger than its maximum size, because this is allowed temporarily during rebalancing activities. Your Auto Scaling group could remain above its maximum size until you resume the `Terminate` process. When `Terminate` resumes, `AZRebalance` gradually rebalances the Auto Scaling group if the group is no longer balanced between Availability Zones or if different Availability Zones are specified.
  - `ReplaceUnhealthy` is inactive but not `HealthCheck`. When `Terminate` resumes, the `ReplaceUnhealthy` process immediately starts running. If any instances were marked as unhealthy while `Terminate` was suspended, they are immediately replaced.

`Launch`

- Your Auto Scaling group does not scale out for alarms or scheduled actions that occur while the process is suspended. `AZRebalance` stops rebalancing the group. `ReplaceUnhealthy` continues

to terminate unhealthy instances, but does not launch replacements. When you resume `Launch`, rebalancing activities and health check replacements are handled in the following way:

- `AZRebalance` gradually rebalances the Auto Scaling group if the group is no longer balanced between Availability Zones or if different Availability Zones are specified.

- `ReplaceUnhealthy` immediately replaces any instances that it terminated during the time that `Launch` was suspended.

AddToLoadBalancer

- Amazon EC2 Auto Scaling launches the instances but does not add them to the load balancer or target group. When you resume the `AddToLoadBalancer` process, it resumes adding instances to the load balancer or target group when they are launched. However, it does not add the instances that were launched while this process was suspended. You must register those instances manually.

AlarmNotification

- Amazon EC2 Auto Scaling does not execute scaling policies when a CloudWatch alarm threshold is in breach. Suspending `AlarmNotification` allows you to temporarily stop scaling events triggered by the group's scaling policies without deleting the scaling policies or their associated CloudWatch alarms. When you resume `AlarmNotification`, Amazon EC2 Auto Scaling considers policies with alarm thresholds that are currently in breach.

AZRebalance

- Your Auto Scaling group does not attempt to redistribute instances after certain events. However, if a scale-out or scale-in event occurs, the scaling process still tries to balance the Availability Zones. For example, during scale out, it launches the instance in the Availability Zone with the fewest instances. If the group becomes unbalanced while `AZRebalance` is suspended and you resume it, Amazon EC2 Auto Scaling attempts to rebalance the group. It first calls `Launch` and then `Terminate`.

HealthCheck

- Amazon EC2 Auto Scaling stops marking instances unhealthy as a result of EC2 and Elastic Load Balancing health checks. Your custom health checks continue to function properly, however. After you suspend `HealthCheck`, if you need to, you can manually set the health state of instances in your group and have `ReplaceUnhealthy` replace them.

ReplaceUnhealthy

- Amazon EC2 Auto Scaling stops replacing instances that are marked as unhealthy. Instances that fail EC2 or Elastic Load Balancing health checks are still marked as unhealthy. As soon as you resume the `ReplaceUnhealthly` process, Amazon EC2 Auto Scaling replaces instances that were marked unhealthy while this process was suspended. The `ReplaceUnhealthy` process calls both of the primary process types—first `Terminate` and then `Launch`.

ScheduledActions

- Amazon EC2 Auto Scaling does not execute scaling actions that are scheduled to run during the suspension period. When you resume `ScheduledActions`, Amazon EC2 Auto Scaling only considers scheduled actions whose execution time has not yet passed.

Suspending Both Launch and Terminate

When you suspend the `Launch` and `Terminate` process types together, the following happens:

- Your Auto Scaling group cannot initiate scaling activities or maintain its desired capacity.

- If the group becomes unbalanced between Availability Zones, Amazon EC2 Auto Scaling does not attempt to redistribute instances evenly between the Availability Zones that are specified for your Auto Scaling group.

- Your Auto Scaling group cannot replace instances that are marked unhealthy.

When you resume the `Launch` and `Terminate` process types, Amazon EC2 Auto Scaling replaces instances that were marked unhealthy while the processes were suspended and might attempt to rebalance the group. Scaling activities also resume.

Additional Considerations

There are some outside operations that might be affected while `Launch` and `Terminate` are suspended.

- **Spot Instance Interruptions**—If `Terminate` is suspended and your Auto Scaling group has Spot Instances, they can still terminate in the event that Spot capacity is no longer available. While `Launch` is suspended, Amazon EC2 Auto Scaling cannot launch replacement instances from another Spot Instance pool or from the same Spot Instance pool when it is available again.

- **Attaching and Detaching Instances** —When `Launch` and `Terminate` are suspended, you can detach instances that are attached to your Auto Scaling group, but you can't attach new instances to the group. To attach instances, you must first resume `Launch`.

    Note
    If detaching an instance will immediately be followed by manually terminating it, you can call the **terminate-instance-in-auto-scaling-group** CLI command instead. This terminates the specified instance and optionally adjusts the group's desired capacity. In addition, if the Auto Scaling group is being used with lifecycle hooks, the custom actions that you specified for instance termination will run before the instance is fully terminated.

- **Standby Instances**—While `Launch` is suspended, you cannot return an instance in the `Standby` state to service. To return the instance to service, you must first resume `Launch`.

Suspend and Resume Scaling Processes (Console)

You can suspend and resume individual processes or all processes.

To suspend and resume processes

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**, choose the process to suspend.

    To resume a suspended process, remove it from **Suspended processes**.

6. Choose **Save**.

Suspend and Resume Scaling Processes (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

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon EC2 Auto Scaling and your AWS solutions. You should collect monitoring data from all parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides monitoring tools to watch Amazon EC2 Auto Scaling, report when something is wrong, and take automatic actions when appropriate.

Health Checks

Amazon EC2 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 groups 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. 160).

Amazon Simple Notification Service Notifications

You can configure Auto Scaling groups to send Amazon SNS notifications when Amazon EC2 Auto Scaling launches or terminates instances. For more information, see Getting Amazon SNS Notifications When Your Auto Scaling Group Scales (p. 162).

CloudWatch Events

Amazon EC2 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 trigger automated actions in other AWS services when these events happen. For more information, see Getting CloudWatch Events When Your Auto Scaling Group Scales (p. 167).

You can also write rules that trigger on API calls made by Amazon EC2 Auto Scaling. For more information, see Creating a CloudWatch Events Rule That Triggers on an AWS API Call Using AWS CloudTrail in the *Amazon CloudWatch Events User Guide*.

CloudWatch Alarms

To detect unhealthy application behavior, CloudWatch helps you by automatically monitoring certain metrics for your AWS resources. You can configure a CloudWatch alarm and set up an Amazon SNS notification that sends an email when a metric's value is not what you expect or when certain anomalies are detected. For example, you can be notified when network activity is suddenly higher or lower than a metric's expected value. For more information, see the Amazon CloudWatch User Guide.

CloudWatch Dashboards

CloudWatch dashboards are customizable home pages in the CloudWatch console. You can use these pages to monitor your resources in a single view, even including resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources. For more information, see the Amazon CloudWatch User Guide.

CloudTrail Logs

AWS CloudTrail enables you to track the calls made to the Amazon EC2 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. Logs

include 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 Amazon EC2 Auto Scaling API Calls with AWS CloudTrail (p. 179).

CloudWatch Logs

CloudWatch Logs enable you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.

AWS Personal Health Dashboard

The Personal Health Dashboard (PHD) displays information, and also provides notifications that are triggered by changes in the health of AWS resources. The information is presented in two ways: on a dashboard that shows recent and upcoming events organized by category, and in a full event log that shows all events from the past 90 days. For more information, see Personal Health Dashboard Notifications for Amazon EC2 Auto Scaling (p. 181).

Health Checks for Auto Scaling Instances

The health status of an Auto Scaling instance is either healthy or unhealthy. All instances in your Auto Scaling group start in the healthy state. Instances are assumed to be healthy unless Amazon EC2 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 a custom health check.

After Amazon EC2 Auto Scaling marks an instance as unhealthy, it is scheduled for replacement. If you do not want instances to be replaced, you can suspend the health check process for any individual Auto Scaling group.

Instance Health Status

Amazon EC2 Auto Scaling can determine the health status of an instance using one or more of the following:

- Status checks provided by Amazon EC2 to identify hardware and software issues that may impair an instance. This includes both instance status checks and system status checks. For more information, see Types of Status Checks in the *Amazon EC2 User Guide for Linux Instances*.
- Health checks provided by Elastic Load Balancing.
- Your custom health checks.

The EC2 status checks are the default health checks for Amazon EC2 Auto Scaling and do not require any special configuration. You can customize the default health checks conducted by your Auto Scaling group by specifying additional checks, however. For more information, see Adding ELB Health Checks (p. 80) and Using Custom Health Checks (p. 162).

Determining Instance Health

After an instance is fully configured and passes the initial health checks, it is considered healthy by Amazon EC2 Auto Scaling. Amazon EC2 Auto Scaling checks that the instances within the Auto Scaling group are running and in good shape by periodically checking the health state of the instances. When it determines that an instance is unhealthy, it terminates that instance and launches a new one. This helps in maintaining the number of running instances at the minimum number (or desired number, if specified) that you defined.

Amazon EC2 Auto Scaling health checks use the results of the Amazon EC2 status checks to determine the health status of an instance. If the instance is in any state other than `running` or if the system status is `impaired`, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance. This includes when the instance has any of the following states:

- `stopping`
- `stopped`
- `terminating`
- `terminated`

If you attached a load balancer or target group to your Auto Scaling group, you can configure the group to mark an instance as unhealthy when Elastic Load Balancing reports it as `OutOfService`. If connection draining is enabled for your load balancer, Amazon EC2 Auto Scaling waits for in-flight requests to complete or the maximum timeout to expire, whichever comes first, before terminating instances due to a scaling event or health check replacement. If you configure your Auto Scaling group to use the Elastic Load Balancing health checks, Amazon EC2 Auto Scaling determines the health status of the instances by checking both the EC2 status checks and the Elastic Load Balancing health checks. For more information, see Adding ELB Health Checks (p. 80).

If you have custom health checks, you can send the information from your health checks to Amazon EC2 Auto Scaling so that Amazon EC2 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 Amazon EC2 Auto Scaling performs a health check on the instance, it will determine that the instance is unhealthy and then launch a replacement instance.

Health Check Grace Period

Frequently, an Auto Scaling instance that has just come into service needs to warm up before it can pass the health check. Amazon EC2 Auto Scaling waits until the health check grace period ends before checking the health status of the instance. Amazon EC2 status checks and Elastic Load Balancing health checks can complete before the health check grace period expires. However, Amazon EC2 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. If you add a lifecycle hook, the grace period does not start until the lifecycle hook actions are completed and the instance enters the `InService` state.

The health check grace period is in seconds. So if, for example, you specify 300 seconds, you get a five-minute time period.

Replacing Unhealthy Instances

After an instance has been marked unhealthy because 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 **set-instance-health** command) to set the instance's health status back to healthy. If the instance is already terminating, you get an error. 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, **set-instance-health** command) is probably useful only for a suspended group. For more information, see Suspending and Resuming Scaling Processes (p. 154).

Amazon EC2 Auto Scaling creates a new scaling activity for terminating the unhealthy instance and then terminates it. Later, 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. For more information, see Disassociating an Elastic IP Address and Reassociating with a Different Instance and Attaching an Amazon EBS Volume to an Instance in the *Amazon EC2 User Guide for Linux Instances*.

Using Custom Health Checks

If you have your own health check system, you can send the instance's health information directly from your system to Amazon EC2 Auto Scaling using the AWS CLI or an AWS SDK. The following examples show how to use the AWS CLI to configure the health state of an instance and then verify the instance's health state.

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": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-123abc45d",
                    "HealthStatus": "Unhealthy",
                    "LifecycleState": "Terminating"
                },
                ...
            ]
        }
    ]
}
```

Getting Amazon SNS Notifications When Your Auto Scaling Group Scales

It is useful to know when Amazon EC2 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 Amazon EC2 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 plaintext 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. 167).

Contents

SNS Notifications

Amazon EC2 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
`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:region: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-id","Availability Zone":"zone"}
```

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 Amazon EC2 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 that 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 acknowledgment 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.
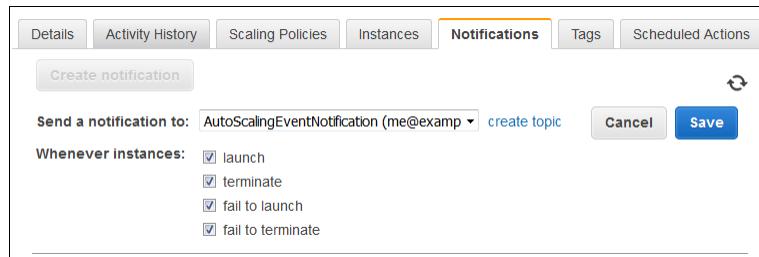
Configure Your Auto Scaling Group to Send Notifications

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 (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 (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. Amazon EC2 Auto Scaling launches the EC2 instance, and you receive an email notification within a few minutes.

To change the desired capacity (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. 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 (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 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 (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 a new instance (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 Amazon EC2 Auto Scaling notification configuration at any time.

To delete Amazon EC2 Auto Scaling notification configuration (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 Amazon EC2 Auto Scaling notification configuration (AWS CLI)

Use the following **delete-notification-configuration** command.

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --topic-
arn arn
```

For information about deleting the Amazon SNS topic and all subscriptions associated with your Auto Scaling group, see Clean Up in the *Amazon Simple Notification Service Developer Guide*.

Getting CloudWatch Events When Your Auto Scaling Group Scales

Amazon CloudWatch Events lets you automate AWS services and respond to system events such as application availability issues or resource changes. Events from AWS services are delivered to CloudWatch Events nearly in real time. You can write simple rules to indicate which events are of interest to you and what automated actions to take when an event matches a rule.

CloudWatch Events lets you set a variety of targets—such as a Lambda function or an Amazon SNS topic —which receive events in JSON format. For more information, see the Amazon CloudWatch Events User Guide.

It is useful to know when Amazon EC2 Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. You can configure Amazon EC2 Auto Scaling to send events to CloudWatch Events whenever your Auto Scaling group scales.

> **Note**
> You can also receive a two-minute warning when Spot Instances are about to be reclaimed by Amazon EC2. For an example of the event for Spot Instance interruption, see Spot Instance Interruption Notices in the *Amazon EC2 User Guide for Linux Instances*.

Contents

Auto Scaling Events

Amazon EC2 Auto Scaling supports sending events to CloudWatch Events when the following events occur:

EC2 Instance-launch Lifecycle Action

Amazon EC2 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-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info"
  }
}
```

EC2 Instance Launch Successful

Amazon EC2 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",
      "instance-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-1234567890abcdef0",
      "StartTime": "yyyy-mm-ddThh:mm:ssZ",
      "Cause": "description-text"
  }
}
```

EC2 Instance Launch Unsuccessful

Amazon EC2 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",
    "instance-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-1234567890abcdef0",
      "StartTime": "yyyy-mm-ddThh:mm:ssZ",
```

```
        "Cause": "description-text"
    }
}
```

EC2 Instance-terminate Lifecycle Action

Amazon EC2 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-1234567890abcdef0",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata":"additional-info"
  }
}
```

EC2 Instance Terminate Successful

Amazon EC2 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",
    "instance-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-1234567890abcdef0",
        "StartTime": "yyyy-mm-ddThh:mm:ssZ",
        "Cause": "description-text"
    }
}
```

EC2 Instance Terminate Unsuccessful

Amazon EC2 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",
    "instance-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-1234567890abcdef0",
      "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, enter `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. Enter 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**, enter a name and description for your rule.

8. 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",
```

Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch

Metrics are the fundamental concept in CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor, and the data points as representing the values of that variable over time. You can use these metrics to verify that your system is performing as expected.

Amazon EC2 Auto Scaling publishes data points to CloudWatch about your Auto Scaling groups. The metrics are available at 1-minute granularity at no additional charge, but you must enable them. By doing this, you get continuous visibility into the operations of your Auto Scaling groups so that you can quickly respond to changes in your workloads. The following sections guide you through enabling them.

Amazon EC2 publishes data points to CloudWatch that describe your Auto Scaling instances. The interval for Amazon EC2 instance monitoring is configurable. You can choose between 1-minute and 5-minute granularity, as described in the sections that follow.

> **Note**
> Amazon EC2 monitoring is available for any EC2 instance, not just those in an Auto Scaling group. For more information, see Monitoring Your Instances Using CloudWatch in the _Amazon EC2 User Guide for Linux Instances_.

Contents

Auto Scaling Group Metrics

The `AWS/AutoScaling` namespace includes the following metrics.

Metric	Description
GroupMinSize	The minimum size of the Auto Scaling group. **Reporting criteria**: Reported if metrics collection is enabled.
GroupMaxSize	The maximum size of the Auto Scaling group.

Metric	Description
	Reporting criteria: Reported if metrics collection is enabled.
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain.

Reporting criteria: Reported if metrics collection is enabled. |
| `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.

Reporting criteria: Reported if metrics collection is enabled. |
| `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.

Reporting criteria: Reported if metrics collection is enabled. |
| `GroupStandbyInstances` | The number of instances that are in a `Standby` state. Instances in this state are still running but are not actively in service.

Reporting criteria: Reported if metrics collection is enabled. |
| `GroupTerminatingInstances` | The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.

Reporting criteria: Reported if metrics collection is enabled. |
| `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.

Reporting criteria: Reported if metrics collection is enabled. |

The `AWS/AutoScaling` namespace includes the following metrics for your groups that use the instance weighting (p. 58) feature. If instance weighting is not applied, then the following metrics are equal to the corresponding metrics defined in the preceding table.

Metric	Description
`GroupInServiceCapacity`	The number of capacity units that are running as part of the Auto Scaling group.

Reporting criteria: Reported if metrics collection is enabled. |
| `GroupPendingCapacity` | The number of capacity units that are pending.

Reporting criteria: Reported if metrics collection is enabled. |
| `GroupStandbyCapacity` | The number of capacity units that are in a `Standby` state.

Reporting criteria: Reported if metrics collection is enabled. |
| `GroupTerminatingCapacity` | The number of capacity units that are in the process of terminating.

Reporting criteria: Reported if metrics collection is enabled. |

Metric	Description
`GroupTotalCapacity`	The total number of capacity units in the Auto Scaling group.
	Reporting criteria: Reported if metrics collection is enabled.

Dimensions for Auto Scaling Group Metrics
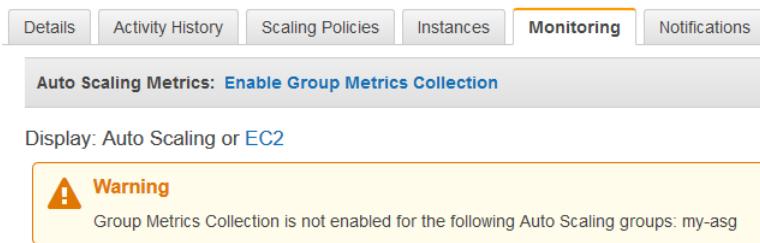
To filter the metrics for your Auto Scaling group by group name, use the `AutoScalingGroupName` dimension.

Enable Auto Scaling Group Metrics

When you enable Auto Scaling group metrics, your Auto Scaling group sends sampled data to CloudWatch every minute. There is no charge for enabling these metrics.
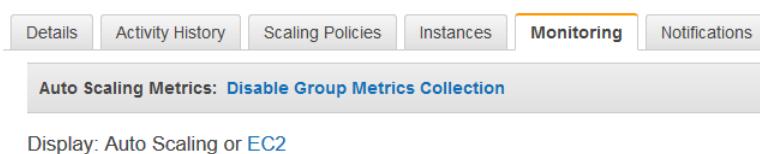
To enable group metrics (console)

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 (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 (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 (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
```

Configure Monitoring for Auto Scaling Instances

You configure monitoring for EC2 instances using a launch configuration or template. Monitoring is enabled whenever an instance is launched, either basic monitoring (5-minute granularity) or detailed monitoring (1-minute granularity). For detailed monitoring, additional charges apply. For more information, see Amazon CloudWatch Pricing.

> **Note**
> By default, basic monitoring is enabled when you create a launch template or when you use the AWS Management Console to create a launch configuration. Detailed monitoring is enabled by default when you create a launch configuration using the AWS CLI or an SDK.

To change the type of monitoring enabled on new EC2 instances, update the launch template or update the Auto Scaling group to use a new launch configuration. Existing instances continue to use the previously enabled monitoring type. To update all instances, terminate them so that they are replaced by your Auto Scaling group or update instances individually using **monitor-instances** and **unmonitor-instances**.

If you have CloudWatch alarms associated with your Auto Scaling group, use the **put-metric-alarm** command to update each alarm. Make each period match 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 period, they continue to check for statistics every minute. They might find no data available for as many as four out of every five periods.

To configure CloudWatch monitoring (console)

When you create the launch configuration using the AWS Management Console, on the **Configure Details** page, select **Enable CloudWatch detailed monitoring**. Otherwise, basic monitoring is enabled. For more information, see Creating a Launch Configuration (p. 36).

To enable detailed monitoring for a launch template using the AWS Management Console, in the **Advanced Details** section, for **Monitoring**, choose **Enable**. Otherwise, basic monitoring is enabled. For more information, see Creating a Launch Template for an Auto Scaling Group (p. 27).

To configure CloudWatch monitoring (AWS CLI)

For launch configurations, 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
```

For launch templates, use the **create-launch-template** command and pass a JSON file that contains the information for creating the launch template. Set the monitoring attribute to `"Monitoring":{"Enabled":true}` to enable detailed monitoring or `"Monitoring":{"Enabled":false}` to enable basic monitoring.

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 Aggregating Statistics by Auto Scaling Group.

To view CloudWatch metrics (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

One purpose for monitoring metrics is to verify that your application is performing as expected. If a metric goes beyond what you consider an acceptable threshold, you can have a CloudWatch alarm trigger an action. You can specify any of the alarm actions that are supported by CloudWatch.

You configure an alarm by identifying the metric to monitor. For example, you can configure an alarm to watch over the average CPU usage of the EC2 instances in your Auto Scaling group. The action can be a notification that is sent to you when the average CPU usage of the group breaches the threshold that you specified for the consecutive periods you specified. For example, if the metric stays at or above 70 percent for 4 consecutive periods of 1 minute each.

For more information, see Using Amazon CloudWatch Alarms in the *Amazon CloudWatch User Guide*.

To create a CloudWatch alarm based on average CPU utilization

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and then **EC2**.
4. Choose a metric category to filter the results. To see the Auto Scaling group metrics, choose **By Auto Scaling Group**. If you only want the metrics for individual instances, choose **Per-Instance Metrics**.
5. Select a metric as follows:

    a. Select the row that contains the Auto Scaling group or instance that you want to create an alarm on and the **CPUUtilization** metric.

    b. Choose the **Graphed metrics** tab.

    c. Under **Statistic**, choose **Average**.

    d. Under **Period**, choose the evaluation period for the alarm, for example, 1 minute. When evaluating the alarm, each period is aggregated into one data point.

    > **Note**
    > A shorter period creates a more sensitive alarm.

    e. Choose **Select metric**.

6. Under **Conditions**, define the alarm by defining the threshold condition. For example, you can define a threshold to trigger the alarm whenever the value of the metric is greater than or equal to 70 percent.

7. Under **Additional configuration**, for **Datapoints to alarm**, specify how many datapoints (evaluation periods) must be in the ALARM state to trigger the alarm, for example, 4 out of 4. This creates an alarm that goes to ALARM state if that many consecutive periods are breaching.

8. For **Missing data treatment**, choose one of the options. For a metric that continually reports data, such as CPUUtilization, you might want to choose **Treat missing data as bad (breaching threshold)**,

as missing datapoints may indicate that something is wrong. For more information, see Configuring How CloudWatch Alarms Treat Missing Data in the *Amazon CloudWatch User Guide*.

9. Choose **Next**.
10. Under **Configure actions**, define the action to take.
11. Choose **Next**.
12. Under **Add a description**, enter a name and description for the alarm and choose **Next**.
13. Choose **Create Alarm**.

Logging Amazon EC2 Auto Scaling API Calls with AWS CloudTrail

Amazon EC2 Auto Scaling is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service using Amazon EC2 Auto Scaling. CloudTrail captures all API calls for Amazon EC2 Auto Scaling as events. The calls captured include calls from the Amazon EC2 Auto Scaling console and code calls to the Amazon EC2 Auto Scaling API.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon EC2 Auto Scaling. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon EC2 Auto Scaling, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon EC2 Auto Scaling Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon EC2 Auto Scaling, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon EC2 Auto Scaling, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon EC2 Auto Scaling actions are logged by CloudTrail and are documented in the Amazon EC2 Auto Scaling API Reference. For example, calls to the **CreateLaunchConfiguration**, **DescribeAutoScalingGroup**, and **UpdateAutoScalingGroup** actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail `userIdentity` Element.

Understanding Amazon EC2 Auto Scaling Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the **CreateLaunchConfiguration** action.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
    },
    "eventTime": "2018-08-21T17:07:49Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "CreateLaunchConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "ebsOptimized": false,
        "instanceMonitoring": {
            "enabled": false
        },
        "instanceType": "t2.micro",
        "keyName": "EC2-key-pair-oregon",
        "blockDeviceMappings": [
            {
                "deviceName": "/dev/xvda",
                "ebs": {
                    "deleteOnTermination": true,
                    "volumeSize": 8,
                    "snapshotId": "snap-01676e0a2c3c7de9e",
                    "volumeType": "gp2"
                }
            }
        ],
        "launchConfigurationName": "launch_configuration_1",
        "imageId": "ami-6cd6f714d79675a5",
```

```
        "securityGroups": [
            "sg-00c429965fd921483"
        ]
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Personal Health Dashboard Notifications for Amazon EC2 Auto Scaling

Your Personal Health Dashboard (PHD) provides support for notifications that come from Amazon EC2 Auto Scaling. These notifications provide awareness and remediation guidance for resource performance or availability issues that may affect your applications. Only events that are specific to missing security groups and launch templates are currently available.

The Personal Health Dashboard is part of the AWS Health service. It requires no setup and can be viewed by any user that is authenticated in your account. For more information, see Getting Started with the AWS Personal Health Dashboard.

If you receive a message similar to the following messages, it should be treated as an alarm to take action.

Example: Auto Scaling group is not scaling out due to a missing security group

```
Hello,

At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
AWS account 123456789012.

A security group associated with this Auto Scaling group cannot be found. Each time a
scale out operation is performed, it will be prevented until you make a change that
fixes the issue.

We recommend that you review and update your Auto Scaling group configuration to change
the launch template or launch configuration that depends on the unavailable security
group.

Sincerely,
Amazon Web Services
```

Example: Auto Scaling group is not scaling out due to a missing launch template

```
Hello,

At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
AWS account 123456789012.

The launch template associated with this Auto Scaling group cannot be found. Each time
a scale out operation is performed, it will be prevented until you make a change that
fixes the issue.

We recommend that you review and update your Auto Scaling group configuration and
specify an existing launch template to use.
```

```
Sincerely,
Amazon Web Services
```

Security in Amazon EC2 Auto Scaling

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon EC2 Auto Scaling, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon EC2 Auto Scaling. The following topics show you how to configure Amazon EC2 Auto Scaling to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon EC2 Auto Scaling resources.

Topics

Amazon EC2 Auto Scaling and Data Protection

Amazon EC2 Auto Scaling conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all of the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields or metadata, such as names and tags. Any data that you enter into metadata might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Encrypting Your Data Using AWS KMS

You can configure your Auto Scaling group to encrypt Amazon EBS volume data stored in the cloud with AWS Key Management Service customer master keys (CMK). Amazon EC2 Auto Scaling supports AWS managed and customer managed CMKs to encrypt your data. Note that the `KmsKeyId` option to specify a customer managed CMK is not available when you use a launch configuration. Use a launch template instead. For more information, see Creating a Launch Template for an Auto Scaling Group (p. 27) and Required CMK Key Policy for Use with Encrypted Volumes (p. 207). You can also use encryption by default to enforce the encryption of the new EBS volumes and snapshot copies that you create.

For more information about protecting your data managed within the Amazon EC2 service, such as EBS volumes and snapshots, see Data Protection in Amazon EC2 and Encryption by Default in the *Amazon EC2 User Guide for Linux Instances*.

For more information about AWS KMS, see What is AWS Key Management Service?

Related topics

- Data Protection in Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*

Identity and Access Management for Amazon EC2 Auto Scaling

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon EC2 Auto Scaling resources. IAM is an AWS service that you can use with no additional charge.

To use Amazon EC2 Auto Scaling, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your AWS account credentials. For more information, see AWS Account Root User Credentials vs. IAM User Credentials in the *AWS General Reference* and IAM Best Practices in the *IAM User Guide*.

For an overview of IAM users and why they are important for the security of your account, see AWS Security Credentials in the *AWS General Reference*.

For details about working with IAM, see the IAM User Guide.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon EC2 Auto Scaling resources. For example, you must have permissions to create Auto Scaling groups, create launch configurations, and so on.

The following sections provide details on how an IAM administrator can use IAM to help secure your Amazon EC2 Auto Scaling resources, by controlling who can perform Amazon EC2 Auto Scaling actions.

We recommend that you read the Amazon EC2 topics first. See Identity and Access Management for Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*. After reading the topics in this section, you should have a good idea what access control permissions Amazon EC2 offers and how they can fit in with your Amazon EC2 Auto Scaling resource permissions.

Topics

- How Amazon EC2 Auto Scaling Works with IAM (p. 185)
- Service-Linked Roles for Amazon EC2 Auto Scaling (p. 189)
- Amazon EC2 Auto Scaling Identity-Based Policy Examples (p. 192)
- Launch Template Support (p. 201)
- IAM Role for Applications That Run on Amazon EC2 Instances (p. 206)
- Required CMK Key Policy for Use with Encrypted Volumes (p. 207)

How Amazon EC2 Auto Scaling Works with IAM

Before you use IAM to manage access to Amazon EC2 Auto Scaling, you should understand what IAM features are available to use with Amazon EC2 Auto Scaling. To get a high-level view of how Amazon EC2 Auto Scaling and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

> **Note**
> Specific IAM permissions and an Amazon EC2 Auto Scaling service-linked role are required so that users can configure Auto Scaling groups.

Topics

- Amazon EC2 Auto Scaling Identity-Based Policies (p. 185)
- Amazon EC2 Auto Scaling Resource-Based Policies (p. 188)
- Access Control Lists (ACLs) (p. 188)
- Authorization Based on Amazon EC2 Auto Scaling Tags (p. 188)
- Amazon EC2 Auto Scaling IAM Roles (p. 188)

Amazon EC2 Auto Scaling Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources, and the conditions under which actions are allowed or denied. Amazon EC2 Auto Scaling supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon EC2 Auto Scaling use the following prefix before the action: `autoscaling:`. Policy statements must include either an `Action` or `NotAction` element. Amazon EC2 Auto Scaling defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as shown in the following example.

```
"Action": [
      "autoscaling:CreateAutoScalingGroup",
```

```
        "autoscaling:UpdateAutoScalingGroup"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "autoscaling:Describe*"
```

To see a list of Amazon EC2 Auto Scaling actions, see Actions in the *Amazon EC2 Auto Scaling API Reference*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

You can restrict access to specific Auto Scaling groups and launch configurations by using their ARNs to identify the resource that the IAM policy applies to. For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

An Auto Scaling group has the following ARN.

```
"Resource":
 "arn:aws:autoscaling:region:123456789012:autoScalingGroup:uuid:autoScalingGroupName/asg-
name"
```

A launch configuration has the following ARN.

```
"Resource":
 "arn:aws:autoscaling:region:123456789012:launchConfiguration:uuid:launchConfigurationName/
lc-name"
```

To specify an Auto Scaling group with the `CreateAutoScalingGroup` action, you must replace the UUID with * as shown in the following.

```
"Resource":
 "arn:aws:autoscaling:region:123456789012:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

To specify a launch configuration with the `CreateLaunchConfiguration` action, you must replace the UUID with * as shown in the following.

```
"Resource":
 "arn:aws:autoscaling:region:123456789012:launchConfiguration:*:launchConfigurationName/lc-
name"
```

Not all Amazon EC2 Auto Scaling actions support resource-level permissions. For actions that don't support resource-level permissions, you must use "*" as the resource.

The following Amazon EC2 Auto Scaling actions do not support resource-level permissions.

- `DescribeAccountLimits`
- `DescribeAdjustmentTypes`
- `DescribeAutoScalingGroups`
- `DescribeAutoScalingInstances`

- `DescribeAutoScalingNotificationTypes`
- `DescribeLaunchConfigurations`
- `DescribeLifecycleHooks`
- `DescribeLifecycleHookTypes`
- `DescribeLoadBalancers`
- `DescribeLoadBalancerTargetGroups`
- `DescribeMetricCollectionTypes`
- `DescribeNotificationConfigurations`
- `DescribePolicies`
- `DescribeScalingActivities`
- `DescribeScalingProcessTypes`
- `DescribeScheduledActions`
- `DescribeTags`
- `DescribeTerminationPolicyTypes`

To learn with which actions you can specify the ARN of each resource, see Actions, Resources, and Condition Keys for Amazon EC2 Auto Scaling in the *IAM User Guide*.

Condition Keys

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the *IAM User Guide*.

Amazon EC2 Auto Scaling defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

The following condition keys are specific to Amazon EC2 Auto Scaling:

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/`*key*
- `autoscaling:SpotPrice`
- `autoscaling:TargetGroupARNs`

- `autoscaling:VPCZoneIdentifiers`

To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon EC2 Auto Scaling in the *IAM User Guide*.

Examples

To view examples of Amazon EC2 Auto Scaling identity-based policies, see Amazon EC2 Auto Scaling Identity-Based Policy Examples (p. 192).

Amazon EC2 Auto Scaling Resource-Based Policies

Amazon EC2 Auto Scaling does not support resource-based policies.

Access Control Lists (ACLs)

Amazon EC2 Auto Scaling does not support Access Control Lists (ACLs).

Authorization Based on Amazon EC2 Auto Scaling Tags

You can apply tag-based, resource-level permissions in the identity-based policies that you create for Amazon EC2 Auto Scaling. This gives you better control over which resources a user can create, modify, use, or delete.

To use tags with IAM policies, you provide tag information in the condition element of a policy using the following condition keys:

- Use `autoscaling:ResourceTag/`*`tag-key`*`:` *`tag-value`* to allow (or deny) user actions on Auto Scaling groups with specific tags.
- Use `aws:RequestTag/`*`tag-key`*`:` *`tag-value`* to require that a specific tag be present (or not present) in a request.
- Use `aws:TagKeys [`*`tag-key`*`, ...]` to require that specific tag keys be present (or not present) in a request.

To view an example policy for controlling access to an Auto Scaling group based on the tags on that Auto Scaling group, see Control Which Scaling Policies Can Be Deleted (p. 196). For other examples for controlling access based on tags, see Amazon EC2 Auto Scaling Identity-Based Policy Examples (p. 192).

Amazon EC2 Auto Scaling IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with Amazon EC2 Auto Scaling

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon EC2 Auto Scaling supports using temporary credentials.

Service-Linked Roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Amazon EC2 Auto Scaling supports service-linked roles. For details about creating or managing Amazon EC2 Auto Scaling service-linked roles, see Service-Linked Roles for Amazon EC2 Auto Scaling (p. 189).

Service Roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. An IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Amazon EC2 Auto Scaling supports service roles for lifecycle hook notifications. For more information, see Amazon EC2 Auto Scaling Lifecycle Hooks (p. 142).

Choosing an IAM Role in Amazon EC2 Auto Scaling

If you have previously created an IAM role that your applications running on Amazon EC2 can assume, you can choose this role when you create a launch template or launch configuration. Amazon EC2 Auto Scaling provides you with a list of roles to choose from. When creating these roles, it's important to associate least privilege IAM policies that restrict access to the specific API calls that the application requires. For more information, see IAM Role for Applications That Run on Amazon EC2 Instances (p. 206).

Service-Linked Roles for Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling uses service-linked roles for the permissions that it requires to call other AWS services on your behalf. A service-linked role is a unique type of IAM role that is linked directly to an AWS service.

Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. For more information, see Using Service-Linked Roles in the *IAM User Guide*. Service-linked roles also enable all API calls to be visible through AWS CloudTrail. This helps with monitoring and auditing requirements because you can track all actions that Amazon EC2 Auto Scaling performs on your behalf. For more information, see Logging Amazon EC2 Auto Scaling API Calls with AWS CloudTrail (p. 179).

The following sections describe how to create and manage Amazon EC2 Auto Scaling service-linked roles. Start by configuring permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Using Service-Linked Roles in the *IAM User Guide*.

Overview

There are two types of Amazon EC2 Auto Scaling service-linked roles:

- The default service-linked role for your account, named **AWSServiceRoleForAutoScaling**. This role is automatically assigned to your Auto Scaling groups unless you specify a different service-linked role.
- A service-linked role with a custom suffix that you specify when you create the role, for example, **AWSServiceRoleForAutoScaling_*mysuffix***.

The permissions of a custom suffix service-linked role are identical to those of the default service-linked role. In both cases, you cannot edit the roles, and you also cannot delete them if they are still in use by an Auto Scaling group. The only difference is the role name suffix.

You can specify either role when you edit your AWS Key Management Service key policies to allow instances that are launched by Amazon EC2 Auto Scaling to be encrypted with your customer managed CMK. However, if you plan to give granular access to a specific customer managed CMK, you should use a custom suffix service-linked role. Using a custom suffix service-linked role provides you with:

- More control over the CMK
- The ability to track which Auto Scaling group made an API call in your CloudTrail logs

If you create customer managed CMKs that not all users should have access to, follow these steps to allow the use of a custom suffix service-linked role:

1. Create a service-linked role with a custom suffix. For more information, see Create a Service-Linked Role (Manual) (p. 191).
2. Give the service-linked role access to a customer managed CMK. For more information about the key policy that allows the CMK to be used by a service-linked role, see Required CMK Key Policy for Use with Encrypted Volumes (p. 207).
3. Give IAM users or roles access to the service-linked role that you created. For more information about creating the IAM policy, see Control Which Service-Linked Role Can Be Passed (Using PassRole) (p. 200). If users try to specify a service-linked role without permission to pass that role to the service, they receive an error.

Permissions Granted by the Service-Linked Role

Amazon EC2 Auto Scaling uses the **AWSServiceRoleForAutoScaling** service-linked role or your custom suffix service-linked role to make the following actions on the specified resources on your behalf:

- `ec2:AttachClassicLinkVpc`
- `ec2:CancelSpotInstanceRequests`
- `ec2:CreateFleet`
- `ec2:CreateTags`
- `ec2:DeleteTags`
- `ec2:Describe*`
- `ec2:DetachClassicLinkVpc`
- `ec2:ModifyInstanceAttribute`
- `ec2:RequestSpotInstances`
- `ec2:RunInstances`
- `ec2:TerminateInstances`
- `elasticloadbalancing:Register*`
- `elasticloadbalancing:Deregister*`
- `elasticloadbalancing:Describe*`
- `cloudwatch:DeleteAlarms`
- `cloudwatch:DescribeAlarms`
- `cloudwatch:PutMetricAlarm`
- `sns:Publish`

The role trusts the `autoscaling.amazonaws.com` service to assume it.

Create a Service-Linked Role (Automatic)

Amazon EC2 Auto Scaling creates the **AWSServiceRoleForAutoScaling** service-linked role for you the first time that you create an Auto Scaling group, unless you manually create a custom suffix service-linked role and specify it when creating the group.

> **Important**
> You must have IAM permissions to create the service-linked role. Otherwise, the automatic creation fails. For more information, see Service-Linked Role Permissions in the *IAM User Guide* and Required Permissions to Create a Service-Linked Role (p. 199) in this guide.

Amazon EC2 Auto Scaling began supporting service-linked roles in March 2018. If you created an Auto Scaling group before then, Amazon EC2 Auto Scaling created the **AWSServiceRoleForAutoScaling** role in your AWS account. For more information, see A New Role Appeared in My AWS Account in the *IAM User Guide*.

Create a Service-Linked Role (Manual)

To create a service-linked role (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, **Create role**.
3. For **Select type of trusted entity**, choose **AWS service**.
4. For **Choose the service that will use this role**, choose **EC2 Auto Scaling** and the **EC2 Auto Scaling** use case.
5. Choose **Next: Permissions**, **Next: Tags**, and then **Next: Review**. Note: You cannot attach tags to service-linked roles during creation.
6. On the **Review** page, leave **Role name** blank to create a service-linked role with the name **AWSServiceRoleForAutoScaling**, or enter a suffix to create a service-linked role with the name **AWSServiceRoleForAutoScaling_*suffix***.
7. (Optional) For **Role description**, edit the description for the service-linked role.
8. Choose **Create role**.

To create a service-linked role (AWS CLI)

Use the following **create-service-linked-role** CLI command to create a service-linked role for Amazon EC2 Auto Scaling with the name **AWSServiceRoleForAutoScaling_*suffix***.

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --custom-
suffix suffix
```

The output of this command includes the ARN of the service-linked role, which you can use to give the service-linked role access to your CMK.

```
{
    "Role": {
        "RoleId": "ABCDEF0123456789ABCDEF",
        "CreateDate": "2018-08-30T21:59:18Z",
        "RoleName": "AWSServiceRoleForAutoScaling_suffix",
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling_suffix",
        "Path": "/aws-service-role/autoscaling.amazonaws.com/",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Principal": {
                        "Service": [
                            "autoscaling.amazonaws.com"
```

```
                            ]
                        },
                        "Effect": "Allow"
                    }
                ]
            }
        }
}
```

For more information, see Creating a Service-Linked Role in the *IAM User Guide*.

Edit the Service-Linked Role

You cannot edit the service-linked roles that are created for Amazon EC2 Auto Scaling. After you create a service-linked role, you cannot change the name of the role or its permissions. However, you can edit the description of the role. For more information, see Editing a Service-Linked Role in the *IAM User Guide*.

Delete the Service-Linked Role

If you are not using an Auto Scaling group, we recommend that you delete its service-linked role. Deleting the role prevents you from having an entity that is not used or actively monitored and maintained.

You can delete a service-linked role only after first deleting the related AWS resources. This protects you from inadvertently revoking Amazon EC2 Auto Scaling permissions to your resources. If a service-linked role is used with multiple Auto Scaling groups, you must delete all Auto Scaling groups that use the service-linked role before you can delete it. For more information, see Deleting Your Auto Scaling Infrastructure (p. 90).

You can use IAM to delete a service-linked role. For more information, see Deleting a Service-Linked Role in the *IAM User Guide.*

If you delete the **AWSServiceRoleForAutoScaling** service-linked role, Amazon EC2 Auto Scaling creates the role again when you create an Auto Scaling group and do not specify a different service-linked role.

Supported Regions for Amazon EC2 Auto Scaling Service-Linked Roles

Amazon EC2 Auto Scaling supports using service-linked roles in all of the AWS Regions where the service is available.

Amazon EC2 Auto Scaling Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Amazon EC2 Auto Scaling resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that give users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics
- Policy Best Practices (p. 193)

The following shows an example of a permissions policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:UpdateAutoScalingGroup",
            "autoscaling:DeleteAutoScalingGroup"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/environment": "test" }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "autoscaling:*LaunchConfiguration*",
            "autoscaling:Describe*"
        ],
        "Resource": "*"
    }]
}
```

This sample policy gives users permissions to create, modify, and delete Auto Scaling groups, but only if the group uses the tag `environment=test`. Because launch configurations do not support tags, and `Describe` actions do not support resource-level permissions, you must specify them in a separate statement without conditions. To learn more about the elements within an IAM policy statement, see Amazon EC2 Auto Scaling Identity-Based Policies (p. 185).

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon EC2 Auto Scaling resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Amazon EC2 Auto Scaling quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.

- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.

- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to

specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

> **Note**
> Some Amazon EC2 Auto Scaling API actions allow you to include specific Auto Scaling groups in your policy that can be created or modified by the action. You can restrict the target resources for these actions by specifying individual Auto Scaling group ARNs. As a best practice, however, we recommend that you use tag-based policies that allow (or deny) actions on Auto Scaling groups with a specific tag.

Predefined AWS Managed Policies

The managed policies that are created by AWS grant the required permissions for common use cases. You can attach these policies to your IAM users, based on the access that they need. Each policy provides access to all or some of the API actions for Amazon EC2 Auto Scaling.

The following are the AWS managed policies for Amazon EC2 Auto Scaling:

- `AutoScalingConsoleFullAccess` — Grants full access to Amazon EC2 Auto Scaling using the AWS Management Console.
- `AutoScalingConsoleReadOnlyAccess` — Grants read-only access to Amazon EC2 Auto Scaling using the AWS Management Console.
- `AutoScalingFullAccess` — Grants full access to Amazon EC2 Auto Scaling.
- `AutoScalingReadOnlyAccess` — Grants read-only access to Amazon EC2 Auto Scaling.

You can also use the `AmazonEC2FullAccess` policy to grant full access to all Amazon EC2 resources and related services, including Amazon EC2 Auto Scaling, CloudWatch, and Elastic Load Balancing.

Customer Managed Policy Examples

You can create your own custom IAM policies to allow or deny permissions for IAM users or groups to perform Amazon EC2 Auto Scaling actions. You can attach these custom policies to the IAM users or groups that require the specified permissions. The following examples show permissions for several common use cases.

If you are new to creating policies, we recommend that you first create an IAM user in your account and attach policies to the user. You can use the console to verify the effects of each policy as you attach the policy to the user.

When creating and updating Auto Scaling groups, some actions require that certain other actions be carried out. You can specify these other actions in the `Action` element of an IAM policy statement. For example, there are additional API actions for Elastic Load Balancing, CloudWatch, and Amazon SNS that might be required depending on the access that you want to provide for a user.

Topics

- Allow Users to Create and Use Launch Templates (p. 199)

Control Which Tag Keys and Tag Values Can Be Used

You can use conditions in your IAM policies to control the tag keys and tag values that can be applied to Auto Scaling groups.

To give users permissions to create or tag an Auto Scaling group only if they specify certain tags, use the `aws:RequestTag` condition key. To allow only specific tag keys, use the `aws:TagKeys` condition key with the `ForAllValues` modifier.

The following policy requires users to specify a tag with the key `environment` in the request. The `"?*"` value enforces that there is some value for the tag key. To use a wildcard, you must use the `StringLike` condition operator.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateOrUpdateTags"
        ],
        "Resource": "*",
        "Condition": {
            "StringLike": { "aws:RequestTag/environment": "?*" }
        }
    }]
}
```

The following policy specifies that users can only tag Auto Scaling groups with the tags `purpose=webserver` and `cost-center=cc123`, and allows only the `purpose` and `cost-center` tags (no other tags can be specified).

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateOrUpdateTags"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/purpose": "webserver",
                "aws:RequestTag/cost-center": "cc123"
            },
            "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
        }
    }]
}
```

The following policy requires users to specify at least one tag in the request, and allows only the `cost-center` and `owner` keys.

```
{
    "Version": "2012-10-17",
    "Statement": [{
```

```
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateOrUpdateTags"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
        }
    }]
}
```

Note
For conditions, the condition key is not case-sensitive and the condition value is case-sensitive. Therefore, to enforce the case-sensitivity of a tag key, use the `aws:TagKeys` condition key, where the tag key is specified as a value in the condition.

Control Access to Auto Scaling Resources Based on Tags

You can also provide tag information in your IAM policies to control access based on the tags that are attached to the Auto Scaling group by using the `autoscaling:ResourceTag` condition key.

Control Access to Creating and Managing Auto Scaling Groups and Scaling Policies

The following policy gives users permissions to use all Amazon EC2 Auto Scaling actions that include the string `Scaling` in their names, as long as the Auto Scaling group has the tag `purpose=webserver`. Because the `Describe` actions do not support resource-level permissions, you must specify them in a separate statement without conditions.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": ["autoscaling:*Scaling*"],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/purpose": "webserver" }
        }
    },
    {
        "Effect": "Allow",
        "Action": "autoscaling:Describe*Scaling*",
        "Resource": "*"
    }]
}
```

Control Which Scaling Policies Can Be Deleted

The following policy allows users to use the `autoscaling:DeletePolicy` action to delete a scaling policy. However, it also denies the action if the Auto Scaling group being acted upon has the tag `environment=production`.

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

```
    },
    {
        "Effect": "Deny",
        "Action": "autoscaling:DeletePolicy",
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
        }
    }]
}
```

Control the Capacity Limits of Auto Scaling Groups

Amazon EC2 Auto Scaling allows you to restrict the size of the Auto Scaling groups that can be created. The following policy gives users permissions to create and update all Auto Scaling groups with the tag `allowed=true`, as long as they don't specify a minimum size less than 1 or a maximum size greater than 10.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:UpdateAutoScalingGroup"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": { "autoscaling:ResourceTag/allowed": "true" },
            "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
            "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
        }
    }]
}
```

Control Which IAM Roles Can Be Passed (Using PassRole)

If you want a user to be able to create an Auto Scaling group using a launch template or launch configuration that specifies an instance profile (a container for an IAM role), you must use a policy that includes a statement allowing the user to pass the role, like the following example. By specifying the ARN, the policy grants the user the permission to pass only roles whose name begins with `qateam-`. For more information, see IAM Role for Applications That Run on Amazon EC2 Instances (p. 206).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qateam-*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Allow Users to Change the Capacity of Auto Scaling Groups

The following policy gives users permissions to use the `SetDesiredCapacity` and `TerminateInstanceInAutoScalingGroup` API actions. The `Resource` element uses a wildcard (*) to indicate that users can change the capacity of any Auto Scaling group.

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

If you are not using tags to control access to Auto Scaling groups, you can adjust the preceding statement to give users permissions to change the capacity of only Auto Scaling groups whose name begins with `devteam-`. For more information about specifying the ARN value, see Resources (p. 186).

```
        "Resource":
 "arn:aws:autoscaling:region:123456789012:autoScalingGroup:*:autoScalingGroupName/devteam-
*"
```

You can also specify multiple ARNs by enclosing them in a list. Including the UUID ensures that access is granted to the specific Auto Scaling group. The UUID for a new group is different than the UUID for a deleted group with the same name.

```
        "Resource": [

 "arn:aws:autoscaling:region:123456789012:autoScalingGroup:7fe02b8e-7442-4c9e-8c8e-85fa99e9b5d9:autoSca
devteam-1",
        "arn:aws:autoscaling:region:123456789012:autoScalingGroup:9d8e8ea4-22e1-44c7-
a14d-520f8518c2b9:autoScalingGroupName/devteam-2",
        "arn:aws:autoscaling:region:123456789012:autoScalingGroup:60d6b363-
ae8b-467c-947f-f1d308935521:autoScalingGroupName/devteam-3"
    ]
```

Allow Users to Create and Use Launch Configurations

The following policy gives users permissions to create a launch configuration if the instance type is `t2.micro`, but only if the name of the launch configuration starts with `qateam-`. For more information about specifying the ARN value, see Resources (p. 186). They can specify a launch configuration for an Auto Scaling group only if its name starts with `qateam-`.

The last part of the statement gives users permissions to describe launch configurations and to access certain Amazon EC2 resources in their AWS account. This gives users minimum permissions to create and manage launch configurations from the Amazon EC2 Auto Scaling console.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": "autoscaling:CreateLaunchConfiguration",
        "Resource":
 "arn:aws:autoscaling:region:123456789012:launchConfiguration:*:launchConfigurationName/
qateam-*",
```

```
            "Condition": {
                "StringEquals": { "autoscaling:InstanceType": "t2.micro" }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": { "autoscaling:LaunchConfigurationName": "qateam-*" }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeLaunchConfigurations",
                "ec2:DescribeImages",
                "ec2:DescribeVolumes",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        }]
}
```

You can add API actions to this policy to provide more options for users, for example:

- `iam:ListInstanceProfiles`: To list instance profiles.
- `ec2:CreateSecurityGroup`: To create a new security group.
- `ec2:AuthorizeSecurityGroupIngress`: To add inbound rules.
- `ec2:CreateKeyPair`: To create a new key pair.

Allow Users to Create and Use Launch Templates

For example policies, see .

Required Permissions to Create a Service-Linked Role

Amazon EC2 Auto Scaling requires permissions to create a service-linked role the first time that any user in your AWS account calls Amazon EC2 Auto Scaling API actions. If the service-linked role does not exist already, Amazon EC2 Auto Scaling creates it in your account. The service-linked role gives permissions to Amazon EC2 Auto Scaling so that it can call other services on your behalf.

For automatic role creation to succeed, users must have permissions for the `iam:CreateServiceLinkedRole` action.

```
"Action": "iam:CreateServiceLinkedRole"
```

The following shows an example of a permissions policy that allows a user to create an Amazon EC2 Auto Scaling service-linked role for Amazon EC2 Auto Scaling.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName":"autoscaling.amazonaws.com"
                }
            }
        }
    ]
}
```

Control Which Service-Linked Role Can Be Passed (Using PassRole)

If your users require the ability to pass custom suffix service-linked roles to an Auto Scaling group, you must attach a policy to the users or roles, based on the access that they need. We recommend that you restrict this policy to only the service-linked roles that your users must access. For more information about custom suffix service-linked roles, see Service-Linked Roles for Amazon EC2 Auto Scaling (p. 189).

The following example is helpful for facilitating the security of your customer managed CMKs if you give different service-linked roles access to different keys. Depending on your needs, you might have a CMK for the development team, another for the QA team, and another for the finance team. First, create a service-linked role that has access to the required CMK, for example, a service-linked role named **AWSServiceRoleForAutoScaling_devteamkeyaccess**. Then, to grant permissions to pass that service-linked role to an Auto Scaling group, attach the policy to your IAM users as shown.

The policy in this example gives users permissions to pass the **AWSServiceRoleForAutoScaling_devteamkeyaccess** role to create any Auto Scaling group whose name begins with devteam-. If they try to specify a different service-linked role, they receive an error. If they choose not to specify a service-linked role, the default **AWSServiceRoleForAutoScaling** role is used instead.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_devteamkeyaccess",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "autoscaling.amazonaws.com"
                    ]
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": [

 "arn:aws:autoscaling:region:123456789012:autoScalingGroup:*:autoScalingGroupName/devteam-
*"
                    ]
                }
            }
        }
    ]
```

```
}
```

Required Permissions for the API

When calling the following actions from the Amazon EC2 Auto Scaling API, users must have permissions from Amazon EC2 and IAM to perform certain actions. You specify the following actions in the `Action` element of an IAM policy statement.

Create an Auto Scaling group using a launch configuration

- `autoscaling:CreateAutoScalingGroup`
- `iam:CreateServiceLinkedRole` (Needed if you are using the default service-linked role and that role does not yet exist)

Create an Auto Scaling group using a launch template

- `autoscaling:CreateAutoScalingGroup`
- `ec2:RunInstances`
- `iam:CreateServiceLinkedRole` (Needed if you are using the default service-linked role and that role does not yet exist)

Update an Auto Scaling group that uses a launch template

- `autoscaling:UpdateAutoScalingGroup`
- `ec2:RunInstances`

Create a launch configuration

- `autoscaling:CreateLaunchConfiguration`
- `ec2:DescribeImages`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeKeyPairs`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeVpcClassicLink`

Launch Template Support

Amazon EC2 Auto Scaling supports using Amazon EC2 launch templates with your Auto Scaling groups. We recommend that you allow users to create Auto Scaling groups from launch templates, because doing so allows them to use the latest features of EC2. In addition, users must specify a launch template to use a mixed instances policy.

You can use the `AmazonEC2FullAccess` policy to give users complete access to work with Amazon EC2 Auto Scaling resources, launch templates, and other EC2 resources in their AWS account. Or, you can create your own custom IAM policies to give users fine-grained permissions to work with specific API actions.

Use the following example policies to give users permissions to work with launch templates, unless another policy already grants them permission to do so. You don't need to give these permissions to

users who are only creating Auto Scaling groups from launch configurations (an older feature of Amazon EC2 Auto Scaling).

For more information about IAM policies for Amazon EC2, see IAM Policies in the *Amazon EC2 User Guide for Linux Instances*.

Topics

- Example: Control Access Using Resource-Level Permissions (p. 199)
- Example: Allow Users to Create and Manage Launch Templates and Launch Template Versions (p. 204)
- Example: Require the Use of Instance Metadata Service Version 2 (IMDSv2) (p. 204)
- Additional Required Permissions (p. 205)

An IAM user or role that creates or updates an Auto Scaling group using a launch template must have permission to use the `ec2:RunInstances` action. Users without this permission receive an error that they are not authorized to use the launch template.

Keep in mind that user permissions for `ec2:RunInstances` are only checked when an Auto Scaling group is created or updated using a launch template. For an Auto Scaling group that is configured to use the `Latest` or `Default` launch template, the permissions are not checked when a new version of the launch template is created. For permissions to be checked, users must configure the Auto Scaling group to use a specific version of the launch template.

Example: Control Access Using Resource-Level Permissions

In granting user permissions, Amazon EC2 allows you to specify resource-level permissions for resources that are created as part of a call to the `ec2:RunInstances` action, to control which resources users can work with. This is a recommended practice.

The following example uses resource-level permissions to restrict access to specific launch templates. It also demonstrates some of the many possible ways that you can control the configuration of an instance that a user can launch when allowing the `ec2:RunInstances` permission.

In this example, there are four statements:

- The first statement restricts user access to launch templates that are located in the specified Region and that have the tag `environment=test`.
- The second statement allows users to tag instances and volumes on creation. This part is needed if there are tags specified in the launch template.
- The third statement requires that users launch instances into a specific subnet (`subnet-1a2b3c4d`), using a specific security group (`sg-1a2b3c4d`), and using a specific AMI (`ami-1a2b3c4d`). It also gives users access to additional resources that they need to launch instances: network interfaces and volumes.
- The fourth statement allows users to launch instances only of a specific instance type (`t2.micro`).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:123456789012:launch-template/*",
            "Condition": {
                "StringEquals": { "ec2:ResourceTag/environment": "test" }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:region:123456789012:*/*",
            "Condition": {
                "StringEquals": { "ec2:CreateAction": "RunInstances" }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region:123456789012:subnet/subnet-1a2b3c4d",
                "arn:aws:ec2:region:123456789012:security-group/sg-1a2b3c4d",
                "arn:aws:ec2:region:123456789012:network-interface/*",
                "arn:aws:ec2:region:123456789012:volume/*",
                "arn:aws:ec2:region::image/ami-1a2b3c4d"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:123456789012:instance/*",
            "Condition": {
                "StringEquals": { "ec2:InstanceType": "t2.micro" }
            }
        }
    ]
}
```

In addition, you can create an IAM policy to limit users' access so that they cannot use launch configurations when creating and updating Auto Scaling groups. Optionally, you can also require that they set the specific version to use, to ensure that the IAM permissions for actions to be completed when launching instances are checked whenever Auto Scaling groups are updated to use a new version.

The following statements give users permissions to create or update an Auto Scaling group if they specify a launch template. If they omit the version number to specify either the `Latest` or `Default` launch template version, or specify a launch configuration instead, the action fails.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": { "autoscaling:LaunchConfigurationName": "false" }
            }
        }
```

```
        }
    ]
}
```

Alternatively, you can require that users specify either the `Latest` or `Default` version of the launch template, rather than a specific version of the template. To do this, change the value of `autoscaling:LaunchTemplateVersionSpecified` to `false`, as shown in the following example.

```
        "Condition": {
            "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "false" }
        }
```

Example: Allow Users to Create and Manage Launch Templates and Launch Template Versions

You can create a policy that grants users permissions to create, modify, describe, and delete launch templates and launch template versions. Before you add these permissions to a user, review the Controlling the Use of Launch Templates section of the *Amazon EC2 User Guide for Linux Instances*. For additional example policies, see Example: Working with Launch Templates in the *Amazon EC2 User Guide for Linux Instances*.

> **Important**
> For groups that are configured to use the `Latest` or `Default` launch template version, permissions for actions to be completed when launching instances are not checked by Amazon EC2 Auto Scaling when a new version of the launch template is created. This is an important consideration when setting up your permissions for who can create and manage launch template versions.

The following policy gives users permissions to create launch templates.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateLaunchTemplate",
            "Resource": "*"
        }
    ]
}
```

Example: Require the Use of Instance Metadata Service Version 2 (IMDSv2)

> **Important**
> If you need to require the use of IMDSv2 on all new instances, your Auto Scaling groups must use launch templates.

For extra security, you can set your users' permissions to require the use of a launch template that requires IMDSv2. For more information, see Configuring the Instance Metadata Service in the *Amazon EC2 User Guide for Linux Instances*.

The following policy specifies that users can't call the `ec2:RunInstances` action unless the instance is also opted in to require the use of IMDSv2 (indicated by `"ec2:MetadataHttpTokens":"required"`). If they do not specify that the instance requires IMDSv2, they get an `UnauthorizedOperation` error when they call the `ec2:RunInstances` action.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
            }
        }
    ]
}
```

To update an existing Auto Scaling group, make sure that it uses a new launch template with the instance metadata options configured, or uses a new version of the current launch template with the instance metadata options configured.

> **Tip**
> To force replacement instances to launch that use your new launch template, you can terminate existing instances in the group. Amazon EC2 Auto Scaling immediately starts launching new instances to replace the instances that you terminated.
> Alternatively, if you use scaling policies, you can increase the desired capacity of the group to launch new instances. If the policy conditions for scale in are met, the Auto Scaling group gradually terminates older instances (depending on the termination policy of the group).

Additional Required Permissions

Depending on which scenarios you want to support, you can specify these additional actions in the `Action` element of an IAM policy statement.

To create and update Auto Scaling groups from the console, users must also have the following permissions from Amazon EC2:

- `ec2:DescribeLaunchTemplates`
- `ec2:DescribeLaunchTemplateVersions`
- `ec2:DescribeVpcs`
- `ec2:DescribeSubnets`
- `ec2:DescribeAvailabilityZones`

Without these additional minimum permissions, launch template and network options cannot load in the Auto Scaling group wizard, and users cannot step through the wizard to launch instances using a launch template.

You can add more actions to your policy to give users more options in the wizard. For example, you can add permissions for Elastic Load Balancing API actions to allow users to select from a list of existing load balancers in Step 3 of the wizard.

Use `iam:PassRole` to allow (or deny) users to pass a role to Amazon EC2 if the launch template specifies an instance profile with an IAM role. Users without this permission receive an error that they are not authorized to use the launch template. For an example policy, see Control Which IAM Roles Can Be Passed (Using PassRole) (p. 197).

To verify provisioning from the Amazon EC2 console, users might need additional permissions (for example, `ec2:DescribeInstances` to view instances, `ec2:DescribeInstanceStatus` to show instance status, or `ec2:DescribeTags` to show tags).

IAM Role for Applications That Run on Amazon EC2 Instances

Applications that run on Amazon EC2 instances need credentials to access other AWS services. To provide these credentials in a secure way, use an IAM role. The role supplies temporary permissions that the application can use when it accesses other AWS resources. The role's permissions determine what the application is allowed to do.

Applications running on the instances can access temporary credentials for the role through the instance profile metadata. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

For instances in an Auto Scaling group, you must create a launch template or launch configuration and choose an instance profile to associate with the instances. An instance profile is a container for an IAM role that allows Amazon EC2 to pass the IAM role to an instance when the instance is launched. First, create an IAM role that has all of the permissions required to access the AWS resources. Then, create the instance profile and assign the role to it. For more information, see Using Instance Profiles in the *IAM User Guide*.

> **Note**
> When you use the IAM console to create a role for Amazon EC2, the console guides you through the steps for creating the role and automatically creates an instance profile with the same name as the IAM role.

For more information, see IAM Roles for Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*.

Prerequisites

Create the IAM role that your application running on Amazon EC2 can assume. Choose the appropriate permissions so that the application that is subsequently given the role can make the specific API calls that it needs.

> **Important**
> As a best practice, we strongly recommend that you create the role so that it has the minimum permissions to other AWS services that your application requires.

To create an IAM role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**, **Create role**.
3. For **Select type of trusted entity**, choose **AWS service**.
4. For **Choose the service that will use this role**, choose **EC2** and the **EC2** use case. Choose **Next: Permissions**.
5. For **Attach permissions policies**, choose the AWS managed policies that contain the required permissions. Choose **Next: Tags** and then **Next: Review**.
6. On the **Review** page, enter a name for the role and choose **Create role**.

When using an instance profile, the `iam:PassRole` permission is needed on the IAM user who creates an Auto Scaling group using a launch template or launch configuration that specifies an instance profile. For an example policy, see Control Which IAM Roles Can Be Passed (Using PassRole) (p. 197).

Create a Launch Configuration

When you create the launch configuration using the AWS Management Console, on the **Configure Details** page, select the role from **IAM role**. For more information, see Creating a Launch Configuration (p. 36).

When you create the launch configuration using the **create-launch-configuration** command from the AWS CLI, specify the name of the instance profile as shown in the following example.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-with-
instance-profile \
--image-id ami-01e24be29428c15b2 --instance-type t2.micro \
--iam-instance-profile my-instance-profile
```

Create a Launch Template

When you create the launch template using the AWS Management Console, in the **Advanced Details** section, select the role from **IAM instance profile**. For more information, see Creating a Launch Template for an Auto Scaling Group (p. 27).

When you create the launch template using the **create-launch-template** command from the AWS CLI, specify the name of the instance profile as shown in the following example.

```
aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --
version-description version1 \
--launch-template-data
 '{"ImageId":"ami-01e24be29428c15b2","InstanceType":"t2.micro","IamInstanceProfile":
{"Name":"my-instance-profile"}}'
```

Required CMK Key Policy for Use with Encrypted Volumes

When creating an encrypted Amazon EBS snapshot or a launch template that specifies encrypted volumes, or enabling encryption by default, you can choose one of the following AWS Key Management Service customer master keys (CMK) to encrypt your data:

- AWS managed CMK — An encryption key in your account that Amazon EBS creates, owns, and manages. This is the default encryption key for a new account. The AWS managed CMK is used for encryption unless you specify a customer managed CMK.
- Customer managed CMK — A custom encryption key that you create, own, and manage. For more information, see Creating Keys in the *AWS Key Management Service Developer Guide*.

  Note: Amazon EBS does not support asymmetric CMKs.

Amazon EC2 Auto Scaling does not need additional authorization to use the default AWS managed CMK to protect the encrypted volumes in your AWS account.

If you specify a customer managed CMK for Amazon EBS encryption, you must give the appropriate service-linked role (p. 189) access to the CMK so that Amazon EC2 Auto Scaling can launch instances on your behalf. To do this, you must modify the CMK's key policy either when the CMK is created or at a later time.

Configuring Key Policies

Use the examples on this page to configure a key policy to give Amazon EC2 Auto Scaling access to your customer managed CMK. You must, at minimum, add two policy statements to your CMK's key policy for it to work with Amazon EC2 Auto Scaling.

- The first statement allows the IAM identity specified in the `Principal` element to use the CMK directly. It includes permissions to perform the AWS KMS `Encrypt`, `Decrypt`, `ReEncrypt*`, `GenerateDataKey*`, and `DescribeKey` operations on the CMK.

- The second statement allows the IAM identity specified in the `Principal` element to use grants to delegate a subset of its own permissions to AWS services that are integrated with AWS KMS or another principal. This allows them to use the CMK to create encrypted resources on your behalf.

When you add the new policy statements to your CMK policy, do not change any existing statements in the policy.

For each of the following examples, arguments that must be replaced, such as a key ID or the name of a service-linked role, are shown as *replaceable text in italics*. In most cases, you can replace the name of the service-linked role with the name of an Amazon EC2 Auto Scaling service-linked role. However, when using a launch configuration to launch Spot Instances, use the role named `AWSServiceRoleForEC2Spot`.

See the following resources:

- To create a CMK with the AWS CLI, see create-key.
- To update a CMK policy with the AWS CLI, see put-key-policy.
- To find a key ID and Amazon Resource Name (ARN), see Finding the Key ID and ARN in the *AWS Key Management Service Developer Guide*.
- For information about Amazon EC2 Auto Scaling service-linked roles, see Service-Linked Roles for Amazon EC2 Auto Scaling (p. 189).

Editing Key Policies in the Console

The examples in the following sections show only how to add statements to a key policy, which is just one way of changing a key policy. The easiest way to change a key policy is to use the IAM console's default view for key policies and make an IAM entity (user or role) one of the *key users* for the appropriate key policy. For more information, see Using the AWS Management Console Default View in the *AWS Key Management Service Developer Guide*.

> **Important**
> Be cautious. The console's default view policy statements include permissions to perform AWS KMS `Revoke` operations on the CMK. If you give an AWS account access to a CMK in your account, and you accidentally revoke the grant that gave them this permission, external users can no longer access their encrypted data or the key that was used to encrypt their data.

Example: CMK Key Policy Sections That Allow Access to the CMK

Add the following two policy statements to the key policy of the customer managed CMK, replacing the example ARN with the ARN of the appropriate service-linked role that is allowed access to the CMK. In this example, the policy sections give the service-linked role named **AWSServiceRoleForAutoScaling** permissions to use the customer managed CMK.

```
{
    "Sid": "Allow service-linked role use of the CMK",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
```

```
    ],
    "Resource": "*"
}
```

```
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
    }
}
```

Example: CMK Key Policy Sections That Allow Cross-Account Access to the CMK

If your customer managed CMK is in a different account than the Auto Scaling group, you must use a grant in combination with the key policy to allow access to the CMK. For more information, see Using Grants in the *AWS Key Management Service Developer Guide*.

First, add the following two policy statements to the CMK's key policy, replacing the example ARN with the ARN of the external account, and specifying the account in which the key can be used. This allows you to use IAM policies to give an IAM user or role in the specified account permission to create a grant for the CMK using the CLI command that follows. Giving the AWS account full access to the CMK does not by itself give any IAM users or roles access to the CMK.

```
{
    "Sid": "Allow external account 111122223333 use of the CMK",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:root"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

```
{
    "Sid": "Allow attachment of persistent resources in external account 111122223333",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
```

```
            "arn:aws:iam::111122223333:root"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*"
}
```

Then, from the external account, create a grant that delegates the relevant permissions to the appropriate service-linked role. The `Grantee Principal` element of the grant is the ARN of the appropriate service-linked role. The `key-id` is the ARN of the CMK. The following is an example create-a-grant CLI command that gives the service-linked role named **AWSServiceRoleForAutoScaling** in account `111122223333` permissions to use the CMK in account `444455556666`.

```
aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

For this command to succeed, the user making the request must have permissions for the `CreateGrant` action. The following example IAM policy allows an IAM user or role in account `111122223333` to create a grant for the CMK in account `444455556666`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow creation of grant for the CMK in external account 444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

If you have any problems configuring the cross-account access to a customer managed CMK that is required to launch an instance with an encrypted volume, see the troubleshooting section (p. 219).

For more information, see Amazon EBS Encryption in the *Amazon EC2 User Guide for Linux Instances* and the AWS Key Management Service Developer Guide.

Compliance Validation for Amazon EC2 Auto Scaling

The security and compliance of Amazon Web Services (AWS) services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Amazon EC2 Auto Scaling is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon EC2 Auto Scaling

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Related topics

- Resilience in Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*

Infrastructure Security in Amazon EC2 Auto Scaling

As a managed service, Amazon EC2 Auto Scaling is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon EC2 Auto Scaling through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Related topics

- Infrastructure Security in Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*

Amazon EC2 Auto Scaling and Interface VPC Endpoints

You can establish a private connection between your virtual private cloud (VPC) and the Amazon EC2 Auto Scaling API by creating an interface VPC endpoint. You can use this connection to call the Amazon EC2 Auto Scaling API from your VPC without sending traffic over the internet. The endpoint provides reliable, scalable connectivity to the Amazon EC2 Auto Scaling API. It does this without requiring an internet gateway, NAT instance, or VPN connection.

Interface VPC endpoints are powered by AWS PrivateLink, a feature that enables private communication between AWS services using private IP addresses. For more information, see AWS PrivateLink.

> **Note**
> You must explicitly enable each API that you want to access through an interface VPC endpoint. For example, you might need to also configure an interface VPC endpoint for `ec2.`*`region`*`.amazonaws.com` to use when calling the Amazon EC2 API operations. For more information, see Interface VPC Endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Create an Interface VPC Endpoint

Create an endpoint for Amazon EC2 Auto Scaling using the following service name:

- **com.amazonaws.*region*.autoscaling** — Creates an endpoint for the Amazon EC2 Auto Scaling API operations.

For more information, see Creating an Interface Endpoint in the *Amazon VPC User Guide*.

Enable private DNS for the endpoint to make API requests to the supported service using its default DNS hostname (for example, `autoscaling.us-east-1.amazonaws.com`). When creating an endpoint for AWS services, this setting is enabled by default. For more information, see Accessing a Service Through an Interface Endpoint in the *Amazon VPC User Guide*.

You do not need to change any Amazon EC2 Auto Scaling settings. Amazon EC2 Auto Scaling calls other AWS services using either public endpoints or private interface VPC endpoints, whichever are in use.

Create a VPC Endpoint Policy

You can attach a policy to your VPC endpoint to control access to the Amazon EC2 Auto Scaling API. The policy specifies:

- The principal that can perform actions.
- The actions that can be performed.
- The resource on which the actions can be performed.

The following example shows a VPC endpoint policy that denies everyone permission to create a scaling policy through the endpoint. The example policy also grants everyone permission to perform all other actions.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
```

```
            "Principal": "*"
        },
        {
            "Action": "autoscaling:PutScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

For more information, see Using VPC Endpoint Policies in the *Amazon VPC User Guide*.

Troubleshooting Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling provides specific and descriptive errors to help you troubleshoot issues. You can find the error messages in the description of the scaling activities.

Contents

Retrieving an Error Message

To retrieve an error message from the description of 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 issues.

EC2 Instance Launch Failures

Issue	Error Message
Auto Scaling group	AutoScalingGroup <Auto Scaling group name> not found. (p. 217)

Issue	Error Message
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request... (p. 218)
AWS account	You are not subscribed to this service. Please see https://aws.amazon.com/. (p. 218)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 218)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 219)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 219)
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)... (p. 218)
Key pair	The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 217)
Launch configuration	The requested configuration is currently not supported. (p. 217)
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 219)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 217)
Service-linked role	Client.InternalError: Client error on launch. (p. 219)

AMI Issues

Issue	Error Message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 221)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 221)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 221)
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. 221)

Load Balancer Issues

Issue	Error Message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 222)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 223)

Issue	Error Message
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 222)
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 223)

Capacity Limits

Issue	Error Message
Capacity limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 224)
Insufficient capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>)... (p. 223)

Troubleshooting Amazon EC2 Auto Scaling: EC2 Instance Launch Failures

This page provides information about your EC2 instances that fail to launch, potential causes, and the steps you can take to resolve the issues.

To retrieve an error message, see Retrieving an Error Message (p. 214).

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. 217)
- The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 217)
- The requested configuration is currently not supported. (p. 217)
- AutoScalingGroup <Auto Scaling group name> not found. (p. 217)
- The requested Availability Zone is no longer supported. Please retry your request... (p. 218)
- Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)... (p. 218)
- You are not subscribed to this service. Please see https://aws.amazon.com/. (p. 218)
- Invalid device name upload. Launching EC2 instance failed. (p. 218)
- Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 219)
- EBS block device mappings not supported for instance-store AMIs. (p. 219)
- Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 219)
- Client.InternalError: Client error on launch. (p. 219)

Amazon EC2 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.

Amazon EC2 Auto Scaling User Guide
The requested Availability Zone is no longer
supported. Please retry your request...

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 https://aws.amazon.com/.

- **Cause**: Your AWS account might have expired.
- **Solution**: Go to https://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**:
  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.

Amazon EC2 Auto Scaling User Guide
Value (<name associated with the instance storage
device>) for parameter virtualName is invalid...

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 a new placement group or launch configuration using the **update-auto-scaling-group** command.

Client.InternalError: Client error on launch.

- **Cause**: This error can be caused when an Auto Scaling group attempts to launch an instance that has an encrypted EBS volume, but the service-linked role does not have access to the customer managed CMK used to encrypt it. For more information, see Required CMK Key Policy for Use with Encrypted Volumes (p. 207).
- **Solution**: Additional setup is required to allow the Auto Scaling group to launch instances. The following table summarizes the steps for resolving the error. For more information, see https://forums.aws.amazon.com/thread.jspa?threadID=277523.

Scenario	Next Steps
Scenario 1: CMK and Auto Scaling group are in the same AWS account	Allow the service-linked role to use the CMK as follows: 1. Determine which service-linked role to use for this Auto Scaling group. 2. Update the key policy on the CMK and allow the service-linked role to use the CMK. 3. Update the Auto Scaling group to use the service-linked role.
Scenario 2: CMK and Auto Scaling group are in different AWS accounts	There are two possible solutions: Solution 1: Use a CMK in the same AWS account as the Auto Scaling group 1. Copy and re-encrypt the snapshot with another CMK that belongs to the same account as the Auto Scaling group. 2. Allow the service-linked role to use the new CMK. See the steps for Scenario 1. Solution 2: Continue to use the CMK in a different AWS account from the Auto Scaling group 1. Determine which service-linked role to use for this Auto Scaling group. 2. Allow the Auto Scaling group account access to the CMK. 3. Define an IAM user or role in the Auto Scaling group account that can create a grant. 4. Create a grant to the CMK with the service-linked role as the grantee principal. 5. Update the Auto Scaling group to use the service-linked role.

Troubleshooting Amazon EC2 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. 214).

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. 221)
- AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 221)
- Value (<ami ID>) for parameter virtualName is invalid. (p. 221)
- 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. 221)

Amazon EC2 Auto Scaling User Guide
The AMI ID <ID of your AMI> does not
exist. Launching EC2 instance failed.

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.

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 Amazon EC2 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. 214).

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. 222)
- There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 222)
- EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 223)
- EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed. (p. 223)
- The security token included in the request is invalid. Validating load balancer configuration failed. (p. 223)

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.

Amazon EC2 Auto Scaling User Guide
EC2 instance <instance ID> is not in VPC.
Updating load balancer configuration failed.

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 https://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. For more information about Amazon EC2 Auto Scaling limits, see Amazon EC2 Auto Scaling Service Quotas (p. 9).

To retrieve an error message, see Retrieving an Error Message (p. 214).

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>)...

- **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

Amazon EC2 Auto Scaling User Guide
<number of instances> instance(s) are already
running. Launching EC2 instance failed.

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 the 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.

- **Amazon EC2 Auto Scaling** – The primary web page for information about Amazon EC2 Auto Scaling.
- **Amazon EC2 Technical FAQ** – The answers to questions customers ask about Amazon EC2 and Amazon EC2 Auto Scaling.
- **Amazon EC2 Discussion Forum** – Get help from the community.
- **AWS Auto Scaling User Guide** – The AWS Auto Scaling console makes it easier for you to use the scaling features of multiple services. With AWS Auto Scaling, you can also simplify the process of defining dynamic scaling policies for your Auto Scaling groups and use predictive scaling to scale your Amazon EC2 capacity in advance of predicted traffic changes.

The following additional resources are available to help you learn more about AWS.

- **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 Amazon EC2 Auto Scaling documentation, beginning in July 2018. For notification about updates to this documentation, you can subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Guide changes (p. 226)	Various improvements in the Replacing Auto Scaling Instances Based on Maximum Instance Lifetime, Auto Scaling Groups with Multiple Instance Types and Purchase Options, Scaling Based on Amazon SQS, and Tagging Auto Scaling Groups and Instances sections of the *Amazon EC2 Auto Scaling User Guide*.	May 6, 2020
Guide changes (p. 226)	Various improvements to IAM documentation. For more information, see Launch Template Support and Amazon EC2 Auto Scaling Identity-Based Policy Examples in the *Amazon EC2 Auto Scaling User Guide*.	March 4, 2020
Disable scaling policies (p. 226)	You can now disable and re-enable scaling policies. This feature allows you to temporarily disable a scaling policy while preserving the configuration details so that you can enable the policy again later. For more information, see Disabling a Scaling Policy for an Auto Scaling Group in the *Amazon EC2 Auto Scaling User Guide*.	February 18, 2020
Add notification functionality (p. 226)	Amazon EC2 Auto Scaling now sends events to your AWS Personal Health Dashboard when your Auto Scaling groups cannot scale out due to a missing security group or launch template. For more information, see Personal Health Dashboard Notifications for Amazon EC2 Auto Scaling in the *Amazon EC2 Auto Scaling User Guide*.	February 12, 2020
Guide changes (p. 226)	Various improvements and corrections in the How Amazon	February 10, 2020

	EC2 Auto Scaling Works with IAM, Amazon EC2 Auto Scaling Identity-Based Policy Examples, Required CMK Key Policy for Use with Encrypted Volumes, and Monitoring Your Auto Scaling Instances and Groups sections of the *Amazon EC2 Auto Scaling User Guide*.	
Guide changes (p. 226)	Improved documentation for Auto Scaling groups that use instance weighting. Learn how to use scaling policies when using "capacity units" to measure desired capacity. For more information, see How Scaling Policies Work and Scaling Adjustment Types in the *Amazon EC2 Auto Scaling User Guide*.	February 6, 2020
New "Security" chapter (p. 226)	A new Security chapter in the *Amazon EC2 Auto Scaling User Guide* helps you understand how to apply the shared responsibility model when using Amazon EC2 Auto Scaling. As part of this update, the user guide chapter "Controlling Access to Your Amazon EC2 Auto Scaling Resources" has been replaced by a new, more useful section, Identity and Access Management for Amazon EC2 Auto Scaling.	February 4, 2020
Recommendations for instance types (p. 226)	AWS Compute Optimizer provides Amazon EC2 instance recommendations to help you improve performance, save money, or both. For more information, see Getting Recommendations for an Instance Type in the *Amazon EC2 Auto Scaling User Guide*.	December 3, 2019
Dedicated Hosts and host resource groups (p. 226)	Updated guide to show how to create a launch template that specifies a host resource group. This allows you to create an Auto Scaling group with a launch template that specifies a BYOL AMI to use on Dedicated Hosts. For more information, see Creating a Launch Template for an Auto Scaling Group in the *Amazon EC2 Auto Scaling User Guide*.	December 3, 2019

Support for Amazon VPC endpoints (p. 226)	You can now establish a private connection between your VPC and Amazon EC2 Auto Scaling. For more information, see Amazon EC2 Auto Scaling and Interface VPC Endpoints in the *Amazon EC2 Auto Scaling User Guide*.	November 22, 2019
Maximum instance lifetime (p. 226)	You can now replace instances automatically by specifying the maximum length of time that an instance can be in service. If any instances are approaching this limit, Amazon EC2 Auto Scaling gradually replaces them. For more information, see Replacing Auto Scaling Instances Based on Maximum Instance Lifetime in the *Amazon EC2 Auto Scaling User Guide*.	November 19, 2019
Instance weighting (p. 226)	For Auto Scaling groups with multiple instance types, you can now optionally specify the number of capacity units that each instance type contributes to the capacity of the group. For more information, see Instance Weighting for Amazon EC2 Auto Scaling in the *Amazon EC2 Auto Scaling User Guide*.	November 19, 2019
Minimum number of instance types (p. 226)	You no longer have to specify additional instance types for groups of Spot, On-Demand, and Reserved Instances. For all Auto Scaling groups, the minimum is now one instance type. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options in the *Amazon EC2 Auto Scaling User Guide*.	September 16, 2019
Support for new Spot allocation strategy (p. 226)	Amazon EC2 Auto Scaling now supports a new Spot allocation strategy "capacity-optimized" that fulfills your request using Spot Instance pools that are optimally chosen based on the available Spot capacity. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options in the *Amazon EC2 Auto Scaling User Guide*.	August 12, 2019

Guide changes (p. 226)	Improved Amazon EC2 Auto Scaling documentation in the Service-Linked Roles and Required CMK Key Policy for Use with Encrypted Volumes topics.	August 1, 2019
Support for tagging enhancement (p. 226)	Amazon EC2 Auto Scaling now adds tags to Amazon EC2 instances as part of the same API call that launches the instances. For more information, see Tagging Auto Scaling Groups and Instances.	July 26, 2019
Guide changes (p. 226)	Improved Amazon EC2 Auto Scaling documentation in the Suspending and Resuming Scaling Processes topic. Updated Customer Managed Policy Examples to include an example policy that allows users to pass only specific custom suffix service-linked roles to Amazon EC2 Auto Scaling.	June 13, 2019
Support for new Amazon EBS feature (p. 226)	Added support for new Amazon EBS feature in the launch template topic. Change the encryption state of an EBS volume while restoring from a snapshot. For more information, see Creating a Launch Template for an Auto Scaling Group in the *Amazon EC2 Auto Scaling User Guide*.	May 13, 2019
Guide changes (p. 226)	Improved Amazon EC2 Auto Scaling documentation in the following sections: Controlling Which Auto Scaling Instances Terminate During Scale In, Auto Scaling Groups, Auto Scaling Groups with Multiple Instance Types and Purchase Options, and Dynamic Scaling for Amazon EC2 Auto Scaling.	March 12, 2019
Support for combining instance types and purchase options (p. 226)	Provision and automatically scale instances across purchase options (Spot, On-Demand, and Reserved Instances) and instance types within a single Auto Scaling group. For more information, see Auto Scaling Groups with Multiple Instance Types and Purchase Options in the *Amazon EC2 Auto Scaling User Guide*.	November 13, 2018

	Updated guide to explain how you can use custom metrics to scale an Auto Scaling group in response to changing demand from an Amazon SQS queue. For more information, see Scaling Based on Amazon SQS in the *Amazon EC2 Auto Scaling User Guide*.	July 26, 2018
Updated topic for scaling based on Amazon SQS (p. 226)		

The following table describes important changes to the Amazon EC2 Auto Scaling documentation before July 2018.

Feature	Description	Release Date
Support for target tracking scaling policies	Set up dynamic scaling for your application in just a few steps. For more information, see Target Tracking Scaling Policies for Amazon EC2 Auto Scaling.	12 July 2017
Support for resource-level permissions	Create IAM policies to control access at the resource level. For more information, see Controlling Access to Your Amazon EC2 Auto Scaling Resources.	15 May 2017
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.	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.	11 August 2016
Events for lifecycle hooks	Amazon EC2 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.	24 February 2016
Instance protection	Prevent Amazon EC2 Auto Scaling from selecting specific instances for termination when scaling in. For more information, see Instance Protection.	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.	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.	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.	19 January 2015

Feature	Description	Release Date
Lifecycle hooks	Hold your newly launched or terminating instances in a pending state while you perform actions on them. For more information, see Amazon EC2 Auto Scaling Lifecycle Hooks.	30 July 2014
Detach instances	Detach instances from an Auto Scaling group. For more information, see Detach EC2 Instances from Your Auto Scaling Group.	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.	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.	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.	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. For information about creating an Auto Scaling group using an EC2 instance, see Creating an Auto Scaling Group Using an EC2 Instance.	02 January 2014
Attach instances	Enable automatic 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.	02 January 2014
View account limits	View the limits on Auto Scaling resources for your account. For more information, see Auto Scaling Limits.	02 January 2014
Console support for Amazon EC2 Auto Scaling	Access Amazon EC2 Auto Scaling using the AWS Management Console. For more information, see Getting Started with Amazon EC2 Auto Scaling.	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.	19 September 2013
Instance termination policy	Specify an instance termination policy for Amazon EC2 Auto Scaling to use when terminating EC2 instances. For more information, see Controlling Which Auto Scaling Instances Terminate During Scale In.	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.	11 June 2012

Feature	Description	Release Date
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.	7 June 2012
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.	26 January 2012
Support for Amazon SNS	Use Amazon SNS to receive notifications whenever Amazon EC2 Auto Scaling launches or terminates EC2 instances. For more information, see Getting SNS Notifications When Your Auto Scaling Group Scales. Amazon EC2 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` operation that tells Amazon EC2 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 operation.	20 July 2011
Scheduled scaling actions	Added support for scheduled scaling actions. For more information, see Scheduled Scaling for Amazon EC2 Auto Scaling.	2 December 2010
Support for Amazon VPC	Added support for Amazon VPC. For more information, see Launching Auto Scaling Instances in a VPC.	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 Amazon EC2 Auto Scaling-managed EC2 instances. For more information, see Using ELB Health Checks with Auto Scaling.	2 December 2010
Support for CloudWatch alarms	Removed the older trigger mechanism and redesigned Amazon EC2 Auto Scaling to use the CloudWatch alarm feature. For more information, see Dynamic Scaling for Amazon EC2 Auto Scaling.	2 December 2010
Suspend and resume scaling	Added support to suspend and resume scaling processes.	2 December 2010

Feature	Description	Release Date
Support for IAM	Added support for IAM. For more information, see Controlling Access to Your Amazon EC2 Auto Scaling Resources.	2 December 2010