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.
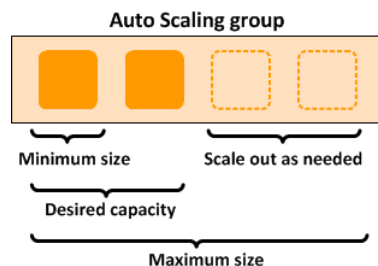
For more information about the benefits of Amazon EC2 Auto Scaling, see Amazon EC2 Auto Scaling benefits (p. 3).

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. 52).
	Configuration templates Your group uses a *launch template*, or a *launch configuration* (not recommended, offers fewer features), 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. 25) and Launch configurations (p. 39).
	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. 111).

Getting started

If you're new to Amazon EC2 Auto Scaling, we recommend that you review Amazon EC2 Auto Scaling instance 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 the Regions and endpoints table 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. You only pay for the AWS resources (for example, EC2 instances and CloudWatch alarms) that you use.

PCI DSS compliance

Amazon EC2 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.

Amazon EC2 Auto Scaling benefits

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.
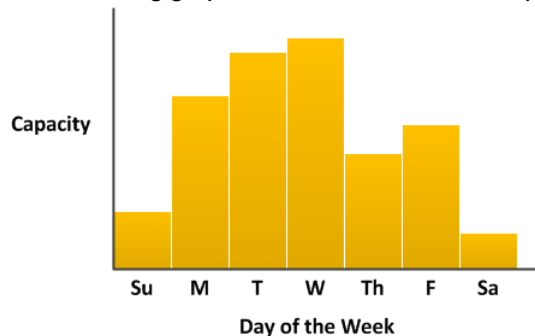
Traditionally, there are two ways to plan for these changes in capacity. The first option is to add enough servers so that the application always has enough capacity to meet demand. The downside of this option, however, is that there are days in which the application doesn't need this much capacity. The extra capacity remains unused and, in essence, raises the cost of keeping the application running.
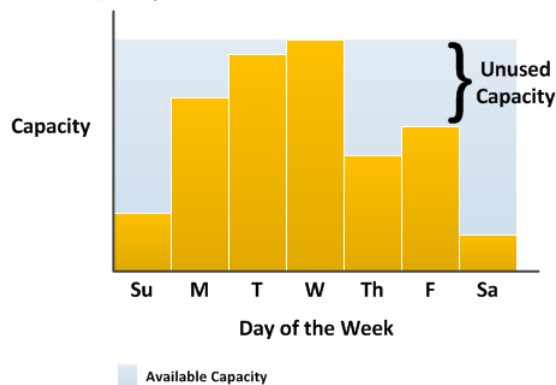
The second option is to have enough capacity to handle the average demand on the application. This option is less expensive, because you aren't purchasing equipment that you'll only use occasionally. However, you risk creating a poor customer experience when the demand on the application exceeds its capacity.
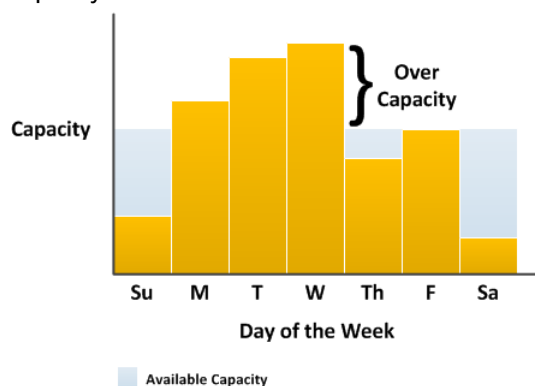
By adding Amazon EC2 Auto Scaling to this application, you have a third option available. You can add new instances to the application only when necessary, and terminate them when they're no longer needed. Because Amazon EC2 Auto Scaling uses EC2 instances, you only have to pay for the instances

you use, when you use them. You now have a cost-effective architecture that provides the best customer experience while minimizing expenses.
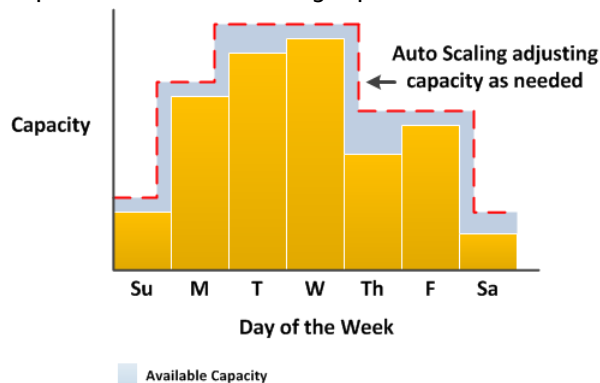
Example: Web app architecture

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.
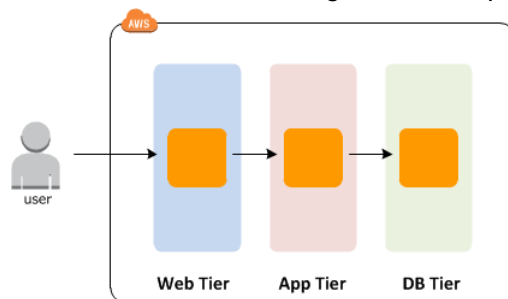
Amazon EC2 Auto Scaling manages the launch and termination of these EC2 instances on your behalf. You define a set of criteria (such as an Amazon CloudWatch alarm) that determines when the Auto Scaling group launches or terminates EC2 instances. Adding Auto Scaling groups to your network architecture helps make your application more highly available and fault tolerant.
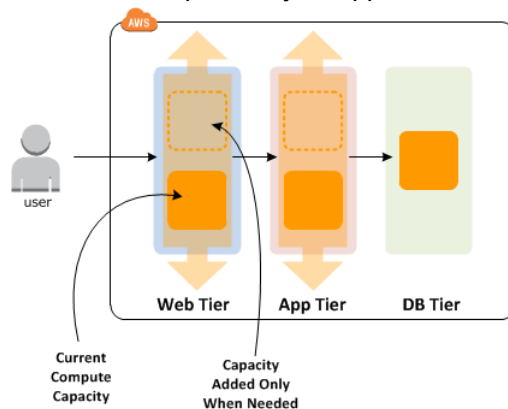
You can create as many Auto Scaling groups as you need. For example, you can create an Auto Scaling group for each tier.

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. 86).

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 the Regions and endpoints table 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.
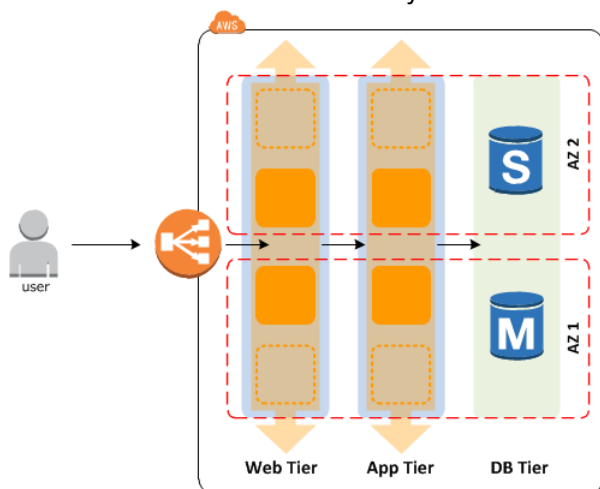
Rebalancing activities

Rebalancing activities fall into two categories: Availability Zone rebalancing and capacity rebalancing.

Availability Zone rebalancing

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).

Capacity Rebalancing

You can enable Capacity Rebalancing for your Auto Scaling groups when using Spot Instances. When you turn on Capacity Rebalancing, Amazon EC2 Auto Scaling attempts to launch a Spot Instance whenever Amazon EC2 notifies that a Spot Instance is at an elevated risk of interruption. After launching a new instance, it then terminates an old instance. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

Amazon EC2 Auto Scaling instance 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.
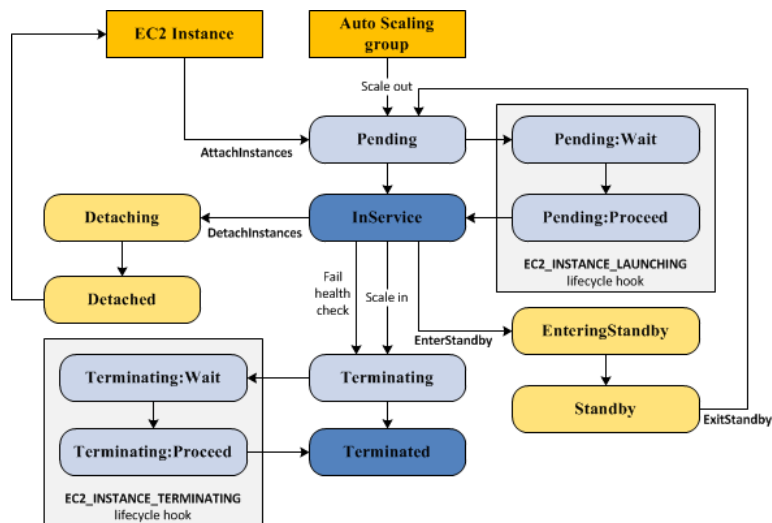
Scale out

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. 114).
- 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. 123).
- 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. 153).

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. 188).
- You put the instance into a `Standby` state. For more information, see Enter and exit standby (p. 10).
- 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. 205).

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. 114).
- 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. 123).
- 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. 153).

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. 116).

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. 120).

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. 158).

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. 195).

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

API-specific limits

- 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.
- You can use `SetInstanceProtection` with at most 50 instances 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

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*.

Install the AWS CLI

To use the AWS CLI with Amazon EC2 Auto Scaling, install the latest AWS CLI version. For information about installing the AWS CLI or upgrading it to the latest version, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

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.

Tasks

Walkthrough summary

In this walkthrough, you:

- Create a configuration template that defines your EC2 instances. You can choose either the launch template or the launch configuration instructions. Although you can use a launch configuration, we recommend a launch template so that you can use the latest features of Amazon EC2 and Amazon EC2 Auto Scaling.
- Create an Auto Scaling group with a single instance in it.
- Terminate the instance and verify that the instance was removed from service and replaced. To maintain a constant number of instances, Amazon EC2 Auto Scaling detects and responds to Amazon EC2 health and reachability checks automatically.

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 an instance type that is 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`.

Preparing for the walkthrough

This walkthrough 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*.

If you are new to Amazon EC2 Auto Scaling and want to get started using the service, you can use the *default* VPC for your AWS account. The default VPC includes a default public subnet in each Availability Zone and an internet gateway that is attached to your VPC. You can view your VPCs on the Your VPCs page of the Amazon Virtual Private Cloud (Amazon VPC) console.

Note that all of the following procedures are for the new console.

Step 1: Create a launch template

In this step, you sign in to the Amazon EC2 console with your AWS account credentials and 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.

> **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.
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. In the left navigation pane, choose **Launch Templates**, and then choose **Create launch template**.
4. For **Launch template name**, enter `my-template-for-auto-scaling`.
5. Under **Auto Scaling guidance**, select the check box.
6. For **Amazon machine image (AMI)**, choose a version of Amazon Linux 2 (HVM) from the **Quick Start** list. The AMI serves as a basic configuration template for your instances.
7. For **Instance type**, choose a hardware configuration that is compatible with the AMI that you specified.

   > **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. (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.
9. Leave **Networking platform** set to **VPC**.

10. For **Security groups**, choose a 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.

11. 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.

12. Choose **Create launch template**.

13. 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 Launch configurations page of the Amazon EC2 console.

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. Choose **Create launch configuration**, and then enter `my-first-launch-configuration` in the **Name** field.

4. For **Amazon machine image (AMI)**, choose an AMI. To find a specific AMI, you can find a suitable AMI, make note of its ID, and enter the ID as search criteria.

   To get the ID of the Amazon Linux 2 AMI:

   a. Open the Amazon EC2 console.

   b. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch instances**.

   c. On the **Quick Start** tab of the **Choose an Amazon Machine Image** page, note the ID of the AMI next to **Amazon Linux 2 AMI (HVM)**. Notice that this AMI is marked "Free tier eligible."

5. For **Instance type**, select a hardware configuration for your 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.

6. Under **Additional configuration**, for **Advanced details**, **IP address type**, make a selection. 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**.

7. For **Security groups**, choose an existing security group. If you leave the **Create a new security group** option selected, a default SSH rule is configured for Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.

8. For **Key pair (login)**, choose an option under **Key pair options** 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.

9. Choose **Create launch configuration**.

10. Select the check box next to the name of your new launch configuration and choose **Actions**, **Create Auto Scaling group**.

Step 2: Create a single-instance Auto Scaling group

Now use Amazon EC2 Auto Scaling to create an Auto Scaling group and add the launch template or launch configuration to the group. Also include information such as the VPC subnets for the instances.

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

To create an Auto Scaling group

1. On the **Choose launch template or configuration** page, for **Auto Scaling group name**, enter `my-first-asg`.
2. 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).
3. [Launch template only] Keep **Purchase options and instance types** set to **Adhere to the launch template**.
4. 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.
5. For **Subnet**, choose a subnet from each Availability Zone that you want to include. Use subnets in multiple Availability Zones for high availability.
6. 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.
7. On the **Review** page, review the information for the group, and then choose **Create Auto Scaling group**.

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.

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

1. Open the Auto Scaling groups page in the Amazon EC2 console.
2. Select the check box next to the Auto Scaling group that you just created.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group. The first tab available is the **Details** tab, showing information about the Auto Scaling group.
3. 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.
4. On the **Instance management** tab, under **Instances**, you can view the status of the instance.

5.   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. Initially, your instance is in the `Pending` state. After an instance is ready to receive traffic, its state is `InService`.

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

Step 4: Terminate an instance in your Auto Scaling group

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.   Open the Auto Scaling groups page in the Amazon EC2 console.
2.   Select the check box next to your Auto Scaling group.
3.   On the **Instance management** tab, under **Instances**, select the ID of the instance.

This takes you to the **Instances** page in the Amazon EC2 console, where you can terminate the instance.
4.   Choose **Actions**, **Instance State**, **Terminate**. When prompted for confirmation, choose **Yes, Terminate**.
5.   On the navigation pane, under **AUTO SCALING**, choose **Auto Scaling Groups**. Select your Auto Scaling group and choose the **Activity** 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.
6.   On the **Instance management** tab, the **Instances** section shows the new instance only.
7.   On the navigation pane, under **INSTANCES**, choose **Instances**. This page shows both the terminated instance and the new running instance.

Step 5: 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 Manual scaling (p. 114).
- Learn how to automatically scale where there are 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. 125).
- Configure an SNS notification to notify you whenever your Auto Scaling group launches or terminates instances. For more information, see Monitoring with Amazon SNS notifications (p. 224).

Step 6: Clean up

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

If you launched an instance that is not within the AWS Free Tier, you should terminate your instance 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 Auto Scaling groups page in the Amazon EC2 console.
2. Select your Auto Scaling group (`my-first-asg`).
3. Choose **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. Open the Amazon EC2 console.
2. On the navigation pane, under **INSTANCES**, choose **Launch Templates**.
3. Select your launch template (`my-template-for-auto-scaling`).
4. 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. Open the Launch configurations page of the Amazon EC2 console.
2. Select your launch configuration (`my-first-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 distribute incoming traffic across your healthy Amazon EC2 instances. This increases the scalability and availability of your application. You can enable Elastic Load Balancing within multiple Availability Zones to increase the fault tolerance of your applications.

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. 87) instead.

Elastic Load Balancing supports different types of load balancers, including Application Load Balancers, Network Load Balancers, and Classic Load Balancers. In most cases, we recommend that you use an Application Load Balancer or a Network Load Balancer. 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 and Amazon EC2 Auto Scaling (p. 86).

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

- (Optional) Create the launch template or launch configuration that will serve as a template for your instances. Choose either the launch template or the launch configuration instructions, based on your preference. Skip this step if you want to use your own launch template or launch configuration.
- Next, create your Auto Scaling group and attach the load balancer.
- Finally, verify that your load balancer is attached.

Contents

Prerequisites

- You have a load balancer and target group created to use. With Application Load Balancers or Network Load Balancers, your Auto Scaling group is registered to a target group that is associated with the load balancer. Make sure that you choose the same Availability Zones for the load balancer that you plan to enable for your Auto Scaling group. For more information, see Getting started with Elastic Load Balancing in the *Elastic Load Balancing User Guide*.
- You have a security group for your launch template or launch configuration that allows access from the load balancer on the listener port (usually port 80 for HTTP traffic) and the port on which you

want Elastic Load Balancing to perform health checks. For more information, see the applicable documentation:

- Target security groups in the *User Guide for Application Load Balancers*
- Target security groups in the *User Guide for Network Load Balancers*
- Security groups for instances in a VPC in the *User Guide for Classic Load Balancers*

If your instances will have public IP addresses, you can also optionally allow SSH traffic if you need to connect to the instances.

- (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, installing software and applications, copying data, and attaching additional EBS volumes. 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.
- This tutorial refers to the default VPC, but you can use your own VPC. In the latter case, make sure that your VPC has a subnet mapped to each Availability Zone of the AWS Region you are working in. At minimum, you must have two public subnets available to create the load balancer and either two private subnets or two public subnets to create your Auto Scaling group and register it with the load balancer.

Deploy your application (console)

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

Tasks

Create a launch template

You must have either a launch template or a launch configuration. If you already have a launch template that you'd like to use, skip this step.

To create a launch template (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 AWS Region where the load balancer was created.
3. On the navigation pane, under **INSTANCES**, choose **Launch Templates**.
4. Choose **Create launch template**.
5. Enter a name and provide a description for the initial version of the launch template.
6. For **Amazon machine image (AMI)**, enter the ID of the AMI for your instances as search criteria.
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 (login)**, choose the key pair to use when connecting to your instances.

9. For **Network interfaces**, do the following:

    a. Choose **Add network interface**. We will use the option to specify the network interface for the purpose of this tutorial.

    b. (Optional) For **Auto-assign public IP**, keep the default value, **Don't include in launch template**. When you create your Auto Scaling group, you can assign a public IP address to instances in your Auto Scaling group by using subnets that have the public IP addressing attribute enabled, such as the default subnets in the default VPC. Alternatively, if you don't need to connect to your instances, you can choose **Disable** to prevent instances in your group from receiving traffic directly from the Internet. In this case, they will receive traffic only from the load balancer.

    c. For **Security group ID**, specify a security group for your instances from the same VPC as the load balancer.

    d. For **Delete on termination**, choose **Yes** so that 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 a launch configuration

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

To create a 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 navigation bar at the top of the screen, choose the AWS Region where the load balancer was created.

4. Choose **Create launch configuration**, and enter a name for your launch configuration.

5. For **Amazon machine image (AMI)**, enter the ID of the AMI for your instances as search criteria.

6. For **Instance type**, select a hardware configuration for your instance.

7. Under **Additional configuration**, pay attention to the following fields:

    a. (Optional) To securely distribute credentials to your EC2 instance, for **IAM instance profile**, select your IAM role. For more information, see IAM role for applications that run on Amazon EC2 instances (p. 259).

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

    c. (Optional) For **Advanced details**, **IP address type**, keep the default value. When you create your Auto Scaling group, you can assign a public IP address to instances in your Auto Scaling group by using subnets that have the public IP addressing attribute enabled, such as the default subnets in the default VPC. Alternatively, if you don't need to connect to your instances, you can choose **Do not assign a public IP address to any instances** to prevent instances in your group from receiving traffic directly from the Internet. In this case, they will receive traffic only from the load balancer.

8. For **Security groups**, choose an existing security group from the same VPC as the load balancer. If you leave the **Create a new security group** option selected, a default SSH rule is configured for Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.

9. For **Key pair (login)**, choose an option under **Key pair options**.

   If you've already configured an Amazon EC2 instance key pair, you can choose it here.

   If you don't already have an Amazon EC2 instance key pair, choose **Create a new key pair** and give it a recognizable name. Choose **Download key pair** to download the key pair to your computer.

   > **Important**
   > If you need to connect to your instances, do not choose **Proceed without a key pair**.

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

11. Select the check box next to the name of your new launch configuration and choose **Actions**, **Create Auto Scaling group**.

Create an Auto Scaling group

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

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

To create an Auto Scaling group (console)

1. If you have not yet navigated to the **Auto Scaling groups** page, do the following:

   - Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

2. Choose **Create an Auto Scaling group**.

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

4. 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.

5. 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).

6. [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.

7. For **Network**, choose the VPC that you used for your load balancer. If you chose the default VPC, it is automatically configured to provide internet connectivity to your instances. This VPC includes a public subnet in each Availability Zone in the Region.

8. For **Subnet**, choose one or more subnets from each Availability Zone that you want to include, based on which Availability Zones the load balancer is in.

9. Choose **Next**.

10. On the **Configure advanced options** page, under **Load balancing**, choose **Enable load balancing**.

11. Do one of the following:

a. Choose **Load balancer target groups**, and then choose the target group for the load balancer you created.

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

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

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

14. 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**.

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. 114).

(Optional) Verify that your load balancer is attached

To verify that your load balancer is attached (console)

1. From the **Auto Scaling groups** page, select the check box next to your Auto Scaling group.

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

3. On the **Instance management** tab, under **Instances**, you can verify that your instances launched successfully. Initially, your instances are in the `Pending` state. After an instance is ready to receive traffic, its state is `InService`. The **Health status** column shows the result of the Amazon EC2 Auto Scaling health checks on your instances. Although an instance may be marked as healthy, the load balancer will only send traffic to instances that pass the load balancer health checks.

4. Verify that your instances are registered with the load balancer. On the navigation pane of the EC2 console, under **LOAD BALANCING**, choose **Target Groups**. Select your target group, and then choose the **Targets** tab. If the state of your instances is `initial`, it's probably because they are still in the process of being registered, or they have not passed the minimum number of health checks to be considered healthy. When the state of your instances is `healthy`, they are ready for use.

Deploy your application (AWS CLI)

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

Tasks
- Create a launch template (p. 22)
- Create a launch configuration (p. 23)
- Create an Auto Scaling group with a load balancer (p. 23)

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.

```
aws ec2 create-launch-template --launch-template-name my-launch-template --version-
description my-version-description \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"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.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-launch-config \
  --image-id ami-01e24be29428c15b2 --instance-type t2.micro --security-groups sg-903004f8
```

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 to automatically increase or decrease the number of instances as the demand on your instances changes. This allows the group to handle changes in the amount of traffic that your application receives. For more information, see Target tracking scaling policies (p. 125).

- If you would like to learn how to create automated schedules for scaling, Scheduled scaling (p. 153) 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. 91).
- 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. 93).

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. 107).

Launch templates

A launch template is similar to a launch configuration (p. 39), 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.

We recommend that you create Auto Scaling groups from launch templates to ensure that you're accessing the latest features and improvements. 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 or multiple launch templates. You must use a launch template to configure these features. For more information, see Auto Scaling groups with multiple instance types and purchase options (p. 53).

In addition to the features of Amazon EC2 Auto Scaling that you can configure by using launch templates, launch templates enable you to use newer features of Amazon EC2. This includes the current generation of EBS volume types (gp3 and io2), EBS volume tagging, T2 Unlimited instances, elastic inference, and Dedicated Hosts, to name a few. 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.

When you create or update an Auto Scaling group with a launch template, you must have sufficient permissions. For more information, see Launch template support (p. 255).

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. 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. 74).

Contents

- Creating your launch template (console) (p. 26)
- Creating a launch template from an existing instance (console) (p. 31)
- Creating a launch template (AWS CLI) (p. 31)

> **Note**
> Keep in mind the following information when creating a launch template for use with an Auto Scaling group:
>
> - A launch template lets you configure additional settings in your Auto Scaling group to launch multiple instance types and combine On-Demand and Spot purchase options, as described in Auto Scaling groups with multiple instance types and purchase options (p. 53). Launching instances with such a combination is not supported:
>   - If you specify a Spot Instance request in the launch template
>   - In EC2-Classic
> - A launch template also enables you to take advantage of newer features of Amazon EC2 such as the current generation of EBS volume types (gp3 and io2), EBS volume tagging, T2 Unlimited instances, elastic inference, and Dedicated Hosts.
> - Support for Dedicated Hosts (host tenancy) is only available if you specify a host resource group. You cannot target a specific host ID or use host placement affinity.
> - A launch template lets you configure a network type (VPC or EC2-Classic), subnet, and Availability Zone. However, these settings are ignored in favor of what is specified in the Auto Scaling group.

Creating your launch template (console)

Follow these steps to configure your launch template for the following:

- Specify the Amazon machine image (AMI) from which to launch the instances.
- Choose an instance type that is compatible with the AMI you've specified.
- Specify the key pair to use when connecting to instances, for example, using SSH.
- Add one or more security groups to allow relevant access to the instances from an external network.
- Specify whether to attach additional EBS volumes or instance store volumes to each instance.
- Add custom tags (key-value pairs) to the instances and volumes.

To create a launch template

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

2. On the navigation pane, under **INSTANCES**, 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 **Auto Scaling guidance**, select the check box to have Amazon EC2 provide guidance to help create a template to use with Amazon EC2 Auto Scaling.

5. Under **Launch template contents**, fill out each required field and any optional fields to use as your instance launch specification.

   a. **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. **Instance type**: Choose the instance type.

   c. (Optional) **Key pair (login)**: Specify a key pair.

6. (Optional) Under **Network settings**, do the following:

   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 security groups as part of the network interface. Each security group must be configured for the VPC that your Auto Scaling group will launch instances into. If you're using EC2-Classic, you must use security groups created specifically for EC2-Classic.

   If you don't specify any security groups in your launch template, Amazon EC2 uses the default security group. By default, this security group doesn't allow inbound traffic from external networks.

7. (Optional) For **Storage (Volumes)**, specify volumes to attach to the instances in addition to the volumes specified by the AMI (**Volume 1 (AMI Root)**). To add a new volume, choose **Add new volume**.

   a. **Volume type**: The type of volume depends on the instance type that you've chosen. Each instance type has an associated root device volume, either an Amazon EBS volume or an instance store volume. 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 for the **Key** parameter.
	Unencrypted snapshot that you own	Encrypted by default CMK	
	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. **Key**: If you chose **Yes** in the previous step, optionally 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. 261) 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.

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

9. To change the default network interface, see Changing the default network interface (p. 29). Skip this step if you want to keep the default network interface (the primary network interface).

10. To configure advanced settings, see Configuring advanced settings for your launch template (p. 30). Otherwise, choose **Create launch template**.

11. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Changing the default network interface

This section shows you how to change the default network interface. This allows you to define, for example, whether you want to assign a public IP address to each instance instead of defaulting to the auto-assign public IP setting on the subnet.

Considerations and limitations

When specifying a network interface, keep in mind the following considerations and limitations:

- 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 security groups in both places.

- You cannot assign specific private IP addresses to your Auto Scaling instances. 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.

- You can launch only one instance if you specify an existing network interface ID. For this to work, you must use the AWS CLI or an SDK to create the Auto Scaling group. When you create the group, you must specify the Availability Zone, but not the subnet ID. Also, you can specify an existing network interface only if it has a device index of 0.

- You cannot auto-assign a public IP address if you specify more than one network interface. You also cannot specify duplicate device indexes across network interfaces. Note that both the primary and secondary network interfaces will reside in the same subnet.

To change the default network interface

1. Under **Network interfaces**, choose **Add network interface**.

2. Specify the primary network interface, paying attention to the following fields:

    a. **Device index**: Specify the device index. Enter 0 for the primary network interface (eth0).

    b. **Network interface**: Leave blank to 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 can 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**: Choose whether to automatically assign a public IP address to the network interface with the device index of 0. This setting takes precedence over settings that you configure for the subnets. If you do not set a value, the default is to use the auto-assign public IP settings of the subnets that your instances are launched into.

    f. **Security groups**: Choose one or more security groups. Each security group must be configured for the VPC that your Auto Scaling group will launch instances into.

    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.

    h. **Elastic Fabric Adapter**: Indicates whether the network interface is an Elastic Fabric Adapter. For more information, see Elastic Fabric Adapter in the *Amazon EC2 User Guide for Linux Instances*.

    i. **Network card index**: Attaches the network interface to a specific network card when using an instance type that supports multiple network cards. The primary network interface (eth0) must be assigned to network card index 0. Defaults to 0 if not specified. For more information, see Network cards in the *Amazon EC2 User Guide for Linux Instances*.

3. To add a secondary network interface, choose **Add network interface**.

Configuring advanced settings for your launch template

You can define any additional capabilities that your Auto Scaling instances need. For example, you can choose an IAM role that your application can use when it accesses other AWS resources or specify the instance user data that can be used to perform common automated configuration tasks after an instance starts.

The following steps discuss the most useful settings to pay attention to. For more information about any of the settings under **Advanced details**, see Creating a launch template in the *Amazon EC2 User Guide for Linux Instances*.

To configure advanced settings

1. For **Advanced details**, expand the section to view the fields.

2. For **Purchasing option**, you can choose **Request Spot Instances** to request Spot Instances at the Spot price, capped at the On-Demand price, and choose **Customize** to change the default Spot Instance settings. For an Auto Scaling group, you must specify a one-time request with no end date (the default). For more information, see Requesting Spot Instances for fault-tolerant and flexible applications (p. 47).

   > **Note**
   > If you leave this setting disabled, you can request Spot Instances later in your Auto Scaling group. This also gives you the option of specifying multiple instance types. That way, if the Amazon EC2 Spot service needs to reclaim your Spot Instances, we can launch replacement instances from another Spot pool. For more information, see Auto Scaling groups with multiple instance types and purchase options (p. 53).

3. For **IAM instance profile**, you can specify an AWS Identity and Access Management (IAM) instance profile to associate with the instances. When you choose an instance profile, you associate the corresponding IAM role with the EC2 instances. For more information, see IAM role for applications that run on Amazon EC2 instances (p. 259).

4. For **Termination protection**, choose whether to protect instances from accidental termination. When you enable termination protection, it provides additional termination protection, but it does not protect from Amazon EC2 Auto Scaling initiated termination. To control whether an Auto Scaling group can terminate a particular instance, use Instance scale-in protection (p. 191).

5. For **Detailed CloudWatch monitoring**, choose whether to enable the instances to publish metric data at 1-minute intervals to Amazon CloudWatch. Additional charges apply. For more information, see Configuring monitoring for Auto Scaling instances (p. 221).

6. For **T2/T3 Unlimited**, choose whether to enable applications to burst beyond the baseline for as long as needed. This field is only valid for T2, T3, and T3a instances. 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*.

7. For **Placement group name**, you can 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.

8. For **Capacity Reservation**, you can specify whether to launch the instances into shared capacity, any `open` Capacity Reservation, a specific Capacity Reservation, or a Capacity Reservation group. For more information, see Launching instances into an existing capacity reservation in the *Amazon EC2 User Guide for Linux Instances*.

9. For **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.

   If you chose **Dedicated Hosts**, complete the following information:

Amazon EC2 Auto Scaling User Guide
Creating a launch template from
an existing instance (console)

- For **Tenancy host resource group**, you can specify a host resource group for a BYOL AMI to use on Dedicated Hosts. 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. Note that an AMI based on a license configuration association can be mapped to only one host resource group at a time. For more information, see Host resource groups in the *AWS License Manager User Guide*.

10. For **License configurations**, specify the license configuration to use. You can launch instances against the specified license configuration to track your license usage. For more information, see Create a license configuration in the *License Manager User Guide*.

11. To configure instance metadata options for all of the instances that are associated with this version of the launch template, do the following:

    a. For **Metadata accessible**, choose whether to enable or disable access to the HTTP endpoint of the instance metadata service. By default, the HTTP endpoint is enabled. If you choose to disable the endpoint, access to your instance metadata is turned off. You can specify the condition to require IMDSv2 only when the HTTP endpoint is enabled.

    b. For **Metadata version**, you can choose to require the use of Instance Metadata Service Version 2 (IMDSv2) when requesting instance metadata. If you do not specify a value, the default is to support both IMDSv1 and IMDSv2.

    c. For **Metadata token response hop limit**, you can set the allowable number of network hops for the metadata token. If you do not specify a value, the default is 1.

    For more information, see Configuring the instance metadata service in the *Amazon EC2 User Guide for Linux Instances*.

12. For **User data**, you can specify user data to configure an instance during launch, or to run a configuration script after the instance starts.

13. Choose **Create launch template**.

14. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Creating a launch template from an existing instance (console)

To create a launch template from an existing instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **INSTANCES**, choose **Instances**.
3. Select the instance and choose **Actions**, **Image and templates**, **Create template from instance**.
4. Provide a name and description.
5. Under **Auto Scaling guidance**, select the check box.
6. Adjust any settings as required, and choose **Create launch template**.
7. To create an Auto Scaling group, choose **Create Auto Scaling group** from the confirmation page.

Creating a launch template (AWS CLI)

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)

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 (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. Select the launch configuration you want to copy and choose **Actions, Copy to launch template**. This sets up a new launch template with the same name and options as the launch configuration that 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. 74).
6. Choose **Copy**.

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. 32).

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. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow automatic scaling to gradually replace older instances with newer instances based on your termination policies (p. 188).

> **Note**
> With the maximum instance lifetime and instance refresh features, you can also replace all instances in the Auto Scaling group to launch new instances that use the launch template. For more information, see Replacing Auto Scaling instances based on maximum instance lifetime (p. 97) and Replacing Auto Scaling instances based on an instance refresh (p. 100).

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. 32).

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. For information about the required IAM permissions, see Launch template support (p. 255).

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

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the page, showing information about the group that's selected.
3. On the **Details** tab, choose **Launch configuration**, **Edit**.
4. Choose **Switch to launch template**.
5. For **Launch template**, select your launch template.
6. For **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 **Update**.

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)

For examples of using a CLI command to update an Auto Scaling group to use a launch template, see Updating an Auto Scaling group to use a launch template (p. 38).

Examples for creating and managing launch templates with the AWS Command Line Interface (AWS CLI)

You can create and manage launch templates through the AWS Management Console, AWS CLI, or SDKs. This section shows you examples of creating and managing launch templates for Amazon EC2 Auto Scaling from the AWS CLI.

Contents
- Creating a simple launch template (p. 34)
- Specifying tags that tag instances on creation (p. 34)
- Specifying an existing network interface (p. 35)
- Assigning public IP addresses (p. 35)

Creating a simple launch template

To create a simple launch template, use the **create-launch-template** command as follows, with these modifications:

- Replace `ami-04d5cc9b88example` with the ID of the AMI from which to launch the instances.
- Replace `t2.micro` with an instance type that is compatible with the AMI that you specified.
- Replace `sg-903004f88example` with a security group for the VPC that your Auto Scaling group will launch instances into.

This example creates a launch template with the name *my-template-for-auto-scaling*. If the instances created by this launch template are launched in a default VPC, they receive a public IP address by default. If the instances are launched in a nondefault VPC, they do not receive a public IP address by default.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","SecurityGroupIds":
["sg-903004f88example"]}'
```

For more information about quoting JSON-formatted parameters, see Using quotation marks with strings in the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can specify the JSON-formatted parameters in a configuration file.

The following example creates a simple launch template, referencing a configuration file for launch template parameter values.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data file://config.json
```

Contents of `config.json`:

```
{
    "ImageId":"ami-04d5cc9b88example",
    "InstanceType":"t2.micro",
    "SecurityGroupIds":["sg-903004f88example"]
}
```

Specifying tags that tag instances on creation

The following example adds a tag (for example, `purpose=webserver`) to instances at launch.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
```

```
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","SecurityGroupIds":
["sg-903004f88example"],"TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"purpose","Value":"webserver"}]}]}'
```

Specifying an existing network interface

The following example configures the primary network interface to use an existing network interface.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"NetworkInterfaceId":"eni-
b9a5ac93","DeleteOnTermination":false}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

Assigning public IP addresses

The following **create-launch-template** example configures the launch template to assign public addresses to instances launched in a nondefault VPC. Note that when you specify a network interface, specify a value for `Groups` that corresponds to security groups for the VPC that your Auto Scaling group will launch instances into, but specify the VPC 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-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","InstanceType":"
```

Adding a secondary network interface

The following **create-launch-template** example adds a secondary network interface. The primary network interface has a device index of 0, and the secondary network interface has a device index of 1.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true},{"DeviceIndex":1,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","InstanceType":"
```

> **Note**
> Attaching multiple network interfaces from the same subnet to an instance can introduce asymmetric routing, especially on instances using a variant of non-Amazon Linux. If you need this type of configuration, you must configure the secondary network interface within the OS. For an example, see How can I make my secondary network interface work in my Ubuntu EC2 instance? in the AWS Knowledge Center.

Creating multiple EFA-enabled network interfaces

If you use an instance type that supports multiple network cards and Elastic Fabric Adapters (EFAs), you can add a secondary interface to a secondary network card and enable EFA using the following **create-launch-template** command. For more information, see Adding an EFA to a launch template in the *Amazon EC2 User Guide for Linux Instances*.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
```

```
  --launch-template-data
 '{"ImageId":"ami-09d95fab7fexample","InstanceType":"p4d.24xlarge","NetworkInterfaces":
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true}]}'
```

> **Warning**
> The p4d.24xlarge instance type incurs higher costs than the other examples in this section. For more information about pricing for P4d instances, see Amazon EC2 P4d Instances pricing.

Specifying a block device mapping

The following **create-launch-template** example creates a launch template with a block device mapping: a 22 gigabyte EBS volume mapped to /dev/xvdcz. The /dev/xvdcz volume uses the General Purpose SSD (gp2) volume type and is deleted when terminating the instance it is attached to.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","SecurityGroupIds":
["sg-903004f88example"],"BlockDeviceMappings":[{"DeviceName":"/dev/xvdcz","Ebs":
{"VolumeSize":22,"VolumeType":"gp2","DeleteOnTermination":true}}]}'
```

Managing your launch templates

The AWS CLI includes several other commands that help you manage your launch templates.

Contents

Listing and describing your launch templates

You can use two AWS CLI commands to get information about your launch templates: **describe-launch-templates** and **describe-launch-template-versions**.

The **describe-launch-templates** command enables you to get a list of any of the launch templates that you have created. You can use an option to filter results on a launch template name, create time, tag key, or tag key/value combination. This command returns summary information about any of your launch templates, including the launch template identifier, latest version, and default version.

The following example provides a summary of the specified launch template.

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

The following is an example response.

```
{
    "LaunchTemplates": [
        {
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
```

```
            "CreateTime": "2020-02-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersionNumber": 1,
            "LatestVersionNumber": 1
        }
    ]
}
```

If you don't use the `--launch-template-names` option to limit the output to one launch template, information on all of your launch templates is returned.

The following **describe-launch-template-versions** command provides information describing the versions of the specified launch template *my-template-for-auto-scaling*.

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

The following is an example response.

```
{
    "LaunchTemplateVersions": [
        {
            "VersionDescription": "version1",
            "LaunchTemplateId": "lt-068f72b729example",
            "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-04d5cc9b88example",
                "InstanceType": "t2.micro",
                "NetworkInterfaces": [
                    {
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "Groups": [
                            "sg-903004f88example"
                        ],
                        "AssociatePublicIpAddress": true
                    }
                ]
            },
            "DefaultVersion": true,
            "CreateTime": "2020-02-28T19:52:27.000Z"
        }
    ]
}
```

Creating a launch template version

The following **create-launch-template-version** command creates a new launch template version based on version 1 of the launch template and specifies a different AMI ID.

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --version-
description version2 \
  --source-version 1 --launch-template-data '{"ImageId":"ami-c998b6b2example"}'
```

To set the default version of the launch template, use the **modify-launch-template** command.

Deleting a launch template version

The following **delete-launch-template-versions** command deletes the specified launch template version.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --
versions 1
```

Deleting a launch template

If you no longer require a launch template, you can delete it using the following **delete-launch-template** command. Deleting a launch template deletes all of its versions.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

Updating an Auto Scaling group to use a launch template

You can use the **update-auto-scaling-group** command to add a launch template to an existing Auto Scaling group.

> **Note**
> If you switch your Auto Scaling group from using a launch configuration, be sure that your permissions are up-to-date. In order to use a launch template, you need specific permissions. For more information, see Launch template support (p. 255).

Updating an Auto Scaling group to use the latest version of a launch template

The following **update-auto-scaling-group** command updates the specified Auto Scaling group to use the latest version of the specified launch template.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

Updating an Auto Scaling group to use a specific version of a launch template

The following **update-auto-scaling-group** command updates the specified Auto Scaling group to use a specific version of the specified launch template.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Launch configurations

Important
We strongly recommend that you do not use launch configurations. They do not provide full
functionality for Amazon EC2 Auto Scaling or Amazon EC2. We provide information about
launch configurations for customers who have not yet migrated from launch configurations to
launch templates.

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. 79). 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. 25).

Contents

Creating a launch configuration

Important
We strongly recommend that you create Auto Scaling groups from launch templates to ensure
that you're getting the latest features from Amazon EC2. For more information, see Creating
a launch template for an Auto Scaling group (p. 26). We provide information about launch
configurations for customers who have not yet migrated from launch configurations to launch
templates.

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. 42).

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. 77).

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. 46).

Contents

- Creating your launch configuration (console) (p. 40)
- Creating a launch configuration (AWS CLI) (p. 41)
- Configuring the instance metadata options (p. 41)

Creating your launch configuration (console)

To create a 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. In the navigation bar, select your AWS Region.
4. Choose **Create launch configuration**, and enter a name for your launch configuration.
5. For **Amazon machine image (AMI)** , choose an AMI. To find a specific AMI, you can find a suitable AMI, make note of its ID, and enter the ID as search criteria.

   To get the ID of the Amazon Linux 2 AMI:

   a. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
   b. In the navigation pane, under **Instances**, choose **Instances**, and then choose **Launch instances**.
   c. On the **Quick Start** tab of the **Choose an Amazon Machine Image** page, note the ID of the AMI next to **Amazon Linux 2 AMI (HVM)**.
6. For **Instance type**, select a hardware configuration for your instances.
7. Under **Additional configuration**, pay attention to the following fields:

   a. (Optional) For **Purchasing option**, you can choose **Request Spot Instances** to request Spot Instances at the Spot price, capped at the On-Demand price. Optionally, you can specify a maximum price per instance hour for your Spot Instances.

      Note
      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. For more information, see Requesting Spot Instances for fault-tolerant and flexible applications (p. 47).
   b. (Optional) For **IAM instance profile**, choose a role to associate with the instances. For more information, see IAM role for applications that run on Amazon EC2 instances (p. 259).
   c. (Optional) For **Monitoring**, choose whether to enable the instances to publish metric data at 1-minute intervals to Amazon CloudWatch by enabling detailed monitoring. Additional charges apply. For more information, see Configuring monitoring for Auto Scaling instances (p. 221).
   d. (Optional) For **Advanced details**, **User data**, you can specify user data to configure an instance during launch, or to run a configuration script after the instance starts.
   e. (Optional) For **Advanced details**, **IP address type**, choose whether to assign a public IP address to the group's instances. If you do not set a value, the default is to use the auto-assign public IP settings of the subnets that your instances are launched into.
8. (Optional) For **Storage (volumes)**, if you don't need additional storage, you can skip this section. Otherwise, to specify volumes to attach to the instances in addition to the volumes specified by the AMI, choose **Add new volume**. Then choose the desired options and associated values for **Devices**, **Snapshot**, **Size**, **Volume type**, **IOPS**, **Throughput**, **Delete on termination**, and **Encrypted**.
9. For **Security groups**, create or select the security group to associate with the group's instances. If you leave the **Create a new security group** option selected, a default SSH rule is configured for

Amazon EC2 instances running Linux. A default RDP rule is configured for Amazon EC2 instances running Windows.

10. For **Key pair (login)**, choose an option under **Key pair options**.

If you've already configured an Amazon EC2 instance key pair, you can choose it here.

If you don't already have an Amazon EC2 instance key pair, choose **Create a new key pair** and give it a recognizable name. Choose **Download key pair** to download the key pair to your computer.

> **Important**
> If you need to connect to your instances, do not choose **Proceed without a key pair**.

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

Creating a launch configuration (AWS CLI)

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)

Configuring the instance metadata options

Amazon EC2 Auto Scaling supports configuring the Instance Metadata Service (IMDS) in launch configurations. This gives you the option of using launch configurations to configure the Amazon EC2 instances in your Auto Scaling groups to require Instance Metadata Service Version 2 (IMDSv2), which is a session-oriented method for requesting instance metadata. For details about IMDSv2's advantages, see this article on the AWS Blog about enhancements to add defense in depth to the EC2 instance metadata service.

You can configure IMDS to support both IMDSv2 and IMDSv1 (the default), or to require the use of IMDSv2. If you are using the AWS CLI or an AWS SDK to configure IMDS, you must use the latest version of the AWS CLI or the SDK to require the use of IMDSv2.

You can configure your launch configuration for the following:

- Require the use of IMDSv2 when requesting instance metadata
- Specify the PUT response hop limit
- Turn off access to instance metadata

You can find more details on configuring the Instance Metadata Service in the following topic: Configuring the instance metadata service in the *Amazon EC2 User Guide for Linux Instances*.

Use the following procedure to configure IMDS options in a launch configuration. After you create your launch configuration, you can associate it with your Auto Scaling group. If you associate the launch configuration with an existing Auto Scaling group, the existing launch configuration is disassociated from the Auto Scaling group, and existing instances will require replacement to use the IMDS options that you specified in the new launch configuration. For more information, see Changing the launch configuration for an Auto Scaling group (p. 46).

To configure IMDS in a 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. In the navigation bar, select your AWS Region.

4. Choose **Create launch configuration**, and create the launch configuration the usual way. Include the ID of the Amazon Machine Image (AMI), the instance type, and optionally, a key pair, one or more security groups, and any additional EBS volumes or instance store volumes for your instances.

5. To configure instance metadata options for all of the instances associated with this launch configuration, in **Additional configuration**, under **Advanced details**, do the following:

    a. For **Metadata accessible**, choose whether to enable or disable access to the HTTP endpoint of the instance metadata service. By default, the HTTP endpoint is enabled. If you choose to disable the endpoint, access to your instance metadata is turned off. You can specify the condition to require IMDSv2 only when the HTTP endpoint is enabled.

    b. For **Metadata version**, you can choose to require the use of Instance Metadata Service Version 2 (IMDSv2) when requesting instance metadata. If you do not specify a value, the default is to support both IMDSv1 and IMDSv2.

    c. For **Metadata token response hop limit**, you can set the allowable number of network hops for the metadata token. If you do not specify a value, the default is 1.

6. When you have finished, choose **Create launch configuration**.

To require the use of IMDSv2 in a launch configuration using the AWS CLI

Use the following **create-launch-configuration** command with `--metadata-options` set to `HttpTokens=required`. When you specify a value for `HttpTokens`, you must also set `HttpEndpoint` to enabled. Because the secure token header is set to required for metadata retrieval requests, this opts in the instance to require using IMDSv2 when requesting instance metadata.

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imdsv2 \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
      ...
  --metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

To turn off access to instance metadata

Use the following **create-launch-configuration** command to turn off access to instance metadata. You can turn access back on later by using the modify-instance-metadata-options command.

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imds-disabled \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
      ...
  --metadata-options "HttpEndpoint=disabled"
```

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.

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, instance monitoring, placement tenancy, ramdisk, security groups, Spot (max) price, user data, whether the instance has a public IP address, and whether the instance is EBS-optimized.

> **Tip**
> You can create an Auto Scaling group directly from an EC2 instance (p. 79). When you use this feature, Amazon EC2 Auto Scaling automatically creates a launch configuration for you as well.

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

Examples

- Create a launch configuration using an EC2 instance (p. 43)
- Create a launch configuration from an instance and override the block devices (AWS CLI) (p. 44)
- Create a launch configuration and override the instance type (AWS CLI) (p. 45)

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. 116).

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": [
```

Amazon EC2 Auto Scaling User Guide
Create a launch configuration from an instance
and override the block devices (AWS CLI)

```
        {
            "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",
            "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",
```

Amazon EC2 Auto Scaling User Guide
Create a launch configuration and
override the instance type (AWS CLI)

```
            "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. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow automatic scaling to gradually replace older instances with newer instances based on your termination policies (p. 188).

> **Note**
> With the maximum instance lifetime and instance refresh features, you can also replace all instances in the Auto Scaling group to launch new instances that use the new launch configuration. For more information, see Replacing Auto Scaling instances based on maximum instance lifetime (p. 97) and Replacing Auto Scaling instances based on an instance refresh (p. 100).

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. On the navigation pane, under **AUTO SCALING**, 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 navigation pane, under **AUTO SCALING**, choose **Auto Scaling Groups**.
6. Select the check box next to the Auto Scaling group.

A split pane opens up in the bottom part of the page, showing information about the group that's selected.

7. On the **Details** tab, choose **Launch configuration**, **Edit**.

8. For **Launch configuration**, select the new launch configuration.

9. When you have finished, choose **Update**.

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.

Requesting Spot Instances for fault-tolerant and flexible applications

Amazon EC2 Spot Instances are spare capacity available at steep discounts compared to the EC2 On-Demand price. You can use Spot Instances for various fault-tolerant and flexible applications.

This topic describes how to launch only Spot Instances in your Auto Scaling group by specifying settings in a launch configuration, rather than in the Auto Scaling group itself. The information in this topic also applies to Auto Scaling groups that request Spot Instances with a launch template.

> **Important**
> Spot Instances are typically used to supplement On-Demand Instances. For this scenario, 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 request to launch Spot Instances only after launching a certain number of On-Demand Instances and then continue to launch some combination of On-Demand Instances and Spot Instances as the group scales. For more information, see Auto Scaling groups with multiple instance types and purchase options (p. 53).

Before launching Spot Instances using Amazon EC2 Auto Scaling, we recommend that you become familiar with launching and managing Spot Instances. 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, the Amazon EC2 Spot service 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. 6).

- **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. 188).

Configuring instance tenancy with Amazon EC2 Auto Scaling

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 physical server with EC2 instance capacity fully dedicated to your use, an isolated server with configurations that you can control.

This topic describes how to launch Dedicated Instances in your Auto Scaling group by specifying settings in a launch configuration. For pricing information and to learn more about Dedicated Instances, see the Amazon EC2 dedicated instances product page and Dedicated Instances in the *Amazon EC2 User Guide for Linux Instances*.

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.

> **Important**
> 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. 26). Before launching Dedicated Hosts, we recommend that you become familiar with launching and managing Dedicated Hosts using AWS License Manager. For more information, see the License Manager User Guide.

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.

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. You can specify the instance placement tenancy for your launch configuration as `default` or `dedicated` using the **create-launch-configuration** CLI command with the `--placement-tenancy` option.

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 instances	Dedicated Instances
`default`	shared-tenancy instances	Dedicated Instances
`dedicated`	Dedicated Instances	Dedicated Instances

To create a launch configuration that creates Dedicated Instances (AWS CLI)

Use the following **create-launch-configuration** command to create a launch configuration that 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": "2020-03-08T23:39:49.011Z",
            "BlockDeviceMappings": [],
            "KeyName": null,
            "SecurityGroups": [],
            "LaunchConfigurationName": "my-launch-config",
            "KernelId": null,
            "RamdiskId": null,
            "InstanceType": "m3.medium"
        }
```

```
        ]
    }
```

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.

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. For more information about giving internet access to instances in a VPC, see Accessing the internet in the *Amazon VPC User Guide*.

Contents

Default VPC

If you created your AWS account after December 4, 2013 or you are creating your Auto Scaling group in a new AWS 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*.

Before you can launch your Auto Scaling instances in a nondefault VPC, you must first create your VPC environment. After you create your VPC and subnets, you launch Auto Scaling instances within the subnets. For more information about creating a VPC, see the *Amazon VPC User 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

By default, all instances in the VPC run as shared tenancy instances. Amazon EC2 Auto Scaling also supports Dedicated Instances and Dedicated Hosts. However, support for Dedicate Hosts is only available for Auto Scaling groups that use a launch template. For more information, see Configuring instance tenancy with Amazon EC2 Auto Scaling (p. 48).

More resources for learning about VPCs

Use the following topics to learn more about VPCs and subnets.

- Private subnets in a VPC
  - VPC with public and private subnets (NAT)
  - NAT instances
  - High availability for Amazon VPC NAT instances: An example
- Public subnets in a VPC
  - VPC with a single public subnet
- General VPC information
  - Amazon VPC User Guide
  - VPC peering
  - Elastic network interfaces
  - Securely connect to Linux instances running in a private VPC

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. 205).

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. 111).

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.

If you're new to Auto Scaling groups, start by creating a launch template or a launch configuration and then use it to create an Auto Scaling group in which all instances have the same instance attributes. You can set the following instance attributes by specifying them as part of the launch template or launch configuration: AMI, block devices, key pair, instance type, security groups, user data, EC2 instance monitoring, instance profile, kernel, ramdisk, the tenancy of the instance, whether the instance has a public IP address, and whether the instance is EBS-optimized. The Getting started with Amazon EC2 Auto Scaling (p. 12) tutorial provides a quick introduction to 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 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).
- Define multiple launch templates to allow instances with different CPU architectures (for example, Arm and x86) to launch in the same Auto Scaling group.
- 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.
- Prioritize instance types that can benefit from Savings Plan or Reserved Instance discount pricing.
- Specify how much On-Demand and Spot capacity to launch, and specify an optional On-Demand base portion.
- Define how Amazon EC2 Auto Scaling should distribute your Spot capacity across instance types.
- Enable Capacity Rebalancing. When you turn on Capacity Rebalancing, Amazon EC2 Auto Scaling attempts to launch a Spot Instance whenever the Amazon EC2 Spot service notifies that a Spot Instance is at an elevated risk of interruption. After launching a new instance, it then terminates an old instance. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

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.

Contents

Allocation strategies

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

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 Savings Plans or Reserved Instances. For more information about Savings Plans or Reserved Instances, see the Amazon 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 the following 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.

Alternatively, you can use the `capacity-optimized-prioritized` allocation strategy and then set the order of instance types in the list of launch template overrides from highest to lowest priority (first to last in the list). Amazon EC2 Auto Scaling honors the instance type priorities on a best-effort basis but optimizes for capacity first. This is a good option for workloads where the possibility of disruption must be minimized, but also the preference for certain instance types matters.

`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

Instances distribution	Total number of running instances across purchase options			
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
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 Best practices for EC2 Spot in the *Amazon EC2 User Guide for Linux Instances*. 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 capacity fluctuates independently for each instance type in an Availability Zone, you can often get more compute capacity 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.
- We recommend that you use the `capacity-optimized` allocation strategy. This means that Amazon EC2 Auto Scaling launches instances using Spot pools that are optimally chosen based on the available Spot capacity, which helps you reduce the possibility of a Spot interruption.
- If you choose 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.

If you intend to specify a maximum price, use the AWS CLI or an SDK 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. 26).

You can create an Auto Scaling group using a launch template only if you have permissions to call the `ec2:RunInstances` action. For more information, see Launch template support (p. 255).

Creating an Auto Scaling group with Spot and On-Demand Instances (console)

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

To create an Auto Scaling group with Spot and On-Demand Instances

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
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. Choose **Create an Auto Scaling group**.
4. On the **Choose launch template or configuration** page, do the following:

Amazon EC2 Auto Scaling User Guide
Creating an Auto Scaling group with
Spot and On-Demand Instances (console)

    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**.

5.   On the **Configure settings** page, for **Instance purchase options**, choose **Combine purchase options and instance types**.

6.   Under **Instances distribution**, do the following:

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

    a.   For **On-Demand base capacity**, 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.

    d.   (Optional) For **Prioritize instance types**, select the check box, and then put the instance types in the **Instance types** section in the desired priority order.

    e.   For **Capacity rebalance**, choose whether to enable or disable Capacity Rebalancing. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

7.   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.

8.   (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.

9.   (Optional) To use instance weighting (p. 69), assign each instance type a relative weight that corresponds to how much the instance should count toward the capacity of the Auto Scaling group.

10. 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.

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. Choose **Next**.

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

13. On the **Configure advanced options** page, configure the following options, and then choose **Next**:

    a.   (Optional) To register your EC2 instances with an Elastic Load Balancing (`ELB`) load balancer, choose an existing load balancer or create a new one. For more information, see Elastic Load Balancing and Amazon EC2 Auto Scaling (p. 86). To create a new load balancer, follow the procedure in Create and attach a new Application Load Balancer or Network Load Balancer (p. 89).

    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.

14. 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. 112).

   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. 128).

   c. (Optional) Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance scale-in protection (p. 191).

15. (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. 224).

16. (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. 82).

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

Configuring Spot allocation strategies (AWS CLI)

The following example configurations show how to launch Spot Instances using the different Spot allocation strategies.

> **Note**
> These examples show how to use a configuration file formatted in JSON or YAML. If you use AWS CLI version 1, you must specify a JSON-formatted configuration file. If you use AWS CLI version 2, you can specify a configuration file formatted in either YAML or JSON.

Examples

Example 1: 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 (`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.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 `c5.large` instance type first. The Spot Instances come from the optimal Spot pool in each Availability Zone based on Spot Instance capacity.

JSON

```
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": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

Alternatively, you can use the following **create-auto-scaling-group** command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

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

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

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Example 2: Launch Spot Instances using the `capacity-optimized-prioritized` 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 (`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.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 `c5.large` instance type first. When Amazon EC2 Auto Scaling attempts to fulfill your Spot capacity, it honors the instance type priorities on a best-effort basis, but optimizes for capacity first.

JSON

```
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": "c5.large"
                    },
                    {
                        "InstanceType": "c5a.large"
                    },
                    {
                        "InstanceType": "m5.large"
                    },
                    {
                        "InstanceType": "m5a.large"
                    },
                    {
                        "InstanceType": "c4.large"
                    },
                    {
                        "InstanceType": "m4.large"
                    },
                    {
                        "InstanceType": "c3.large"
                    },
                    {
                        "InstanceType": "m3.large"
                    }
                ]
            },
            "InstancesDistribution": {
                "OnDemandBaseCapacity": 1,
                "OnDemandPercentageAboveBaseCapacity": 0,
                "SpotAllocationStrategy": "capacity-optimized-prioritized"
            }
        },
        "MinSize": 1,
        "MaxSize": 5,
        "DesiredCapacity": 3,
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

Alternatively, you can use the following **create-auto-scaling-group** command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

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

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

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
```

```
      - InstanceType: c5a.large
      - InstanceType: m5.large
      - InstanceType: m5a.large
      - InstanceType: c4.large
      - InstanceType: m4.large
      - InstanceType: c3.large
      - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Example 3: 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 (`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.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 `c5.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.

JSON

```
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": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
```

```
            {
                "InstanceType": "m5a.large"
            },
            {
                "InstanceType": "c4.large"
            },
            {
                "InstanceType": "m4.large"
            },
            {
                "InstanceType": "c3.large"
            },
            {
                "InstanceType": "m3.large"
            }
        ]
    },
    "InstancesDistribution": {
        "OnDemandPercentageAboveBaseCapacity": 50,
        "SpotAllocationStrategy": "lowest-price",
        "SpotInstancePools": 2
    }
},
"MinSize": 1,
"MaxSize": 5,
"DesiredCapacity": 3,
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

Alternatively, you can use the following **create-auto-scaling-group** command to create the Auto Scaling group, referencing a YAML file as the sole parameter for your Auto Scaling group instead of a JSON file.

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

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

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

Amazon EC2 Auto Scaling User Guide
Verifying whether your Auto Scaling
group is configured correctly and that the
group has launched instances (AWS CLI)

Note
For additional examples, see Specifying a different launch template for an instance type (p. 65), Configuring instance weighting for Amazon EC2 Auto Scaling (p. 69), and Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

Verifying whether your Auto Scaling group is configured correctly and that the group has launched instances (AWS CLI)

To check whether your Auto Scaling group is configured correctly and that it has launched instances, use the **describe-auto-scaling-groups** command. Verify that the mixed instances policy and the list of subnets exist and are configured correctly. If the instances have launched, you will see a list of the instances and their statuses. To view the scaling activities that result from instances launching, use the **describe-scaling-activities** command. You can monitor scaling activities that are in progress and that are recently completed.

Configuring overrides

Depending on your requirements, you must override some attributes in the current launch template when using multiple instance types. For example, some instance types might require a launch template with a different AMI. To do so, you must configure the `Overrides` property.

The `Overrides` property allows you to define a set of properties that Amazon EC2 Auto Scaling can use to launch instances, including:

- `InstanceType` — The instance type. For more information about the instance types that are available, see Instance types in the *Amazon EC2 User Guide for Linux Instances*.
- `LaunchTemplateSpecification` — (Optional) A launch template for an individual instance type. This property is currently limited to the AWS CLI or an SDK, and is not available from the console.
- `WeightedCapacity` — (Optional) The number of units that a provisioned instance of this type provides toward fulfilling the desired capacity of the Auto Scaling group.

Contents
- Specifying a different launch template for an instance type (p. 65)
- Configuring instance weighting for Amazon EC2 Auto Scaling (p. 69)

Specifying a different launch template for an instance type

The `Overrides` property allows you to define a new launch template for individual instance types for a new or existing Auto Scaling group. For example, if the architecture of an instance type requires a different AMI from the rest of the group, you must specify a launch template with a compatible AMI.

Say that you configure an Auto Scaling group for compute-intensive applications and want to include a mix of C5, C5a, and C6g instance types. However, C6g instances feature an AWS Graviton processor based on 64-bit Arm architecture, while the C5 and C5a instances run on 64-bit Intel x86 processors. The AMI for C5 instances works on C5a instances and vice-versa, but not on C6g instances. The `Overrides` property allows you to include a different launch template for C6g instances, while still using the same launch template for C5 and C5a instances.

Note
Currently, this feature is available only if you use the AWS CLI or an SDK, and is not available from the console.

Adding or changing a launch template for an instance type (AWS CLI)

The following procedure shows you how to use the AWS CLI to configure an Auto Scaling group so that one or more of the instance types uses a launch template that is different from the rest of the group.

To create and configure a new Auto Scaling group

1. Create a configuration file where you specify a mixed instances policy structure and include the `Overrides` parameter.

   The following are the contents of an example configuration file formatted in JSON. It specifies the `c5.large`, `c5a.large`, and `c6g.large` instance types and defines a new launch template for the `c6g.large` instance type to ensure that an appropriate AMI is used to launch Arm instances. Amazon EC2 Auto Scaling uses the order of instance types to determine which instance type to use first when fulfilling On-Demand capacity.

   ```
   {
       "AutoScalingGroupName":"my-asg",
       "MixedInstancesPolicy":{
           "LaunchTemplate":{
               "LaunchTemplateSpecification":{
                   "LaunchTemplateName":"my-launch-template-for-x86",
                   "Version":"$Latest"
               },
               "Overrides":[
                   {
                       "InstanceType":"c6g.large",
                       "LaunchTemplateSpecification": {
                           "LaunchTemplateName": "my-launch-template-for-arm",
                           "Version": "$Latest"
                       }
                   },
                   {
                       "InstanceType":"c5.large"
                   },
                   {
                       "InstanceType":"c5a.large"
                   }
               ]
           },
           "InstancesDistribution":{
               "OnDemandBaseCapacity": 1,
               "OnDemandPercentageAboveBaseCapacity": 50,
               "SpotAllocationStrategy": "capacity-optimized"
           }
       },
       "MinSize":1,
       "MaxSize":5,
       "DesiredCapacity":3,
       "VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
       "Tags":[ ]
   }
   ```

2. Use the following **create-auto-scaling-group** command, referencing the JSON file as the sole parameter for your Auto Scaling group.

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

To change the launch template for an instance type in an existing Auto Scaling group

- Use the following **update-auto-scaling-group** command to specify a different launch template for an instance type by passing the `Overrides` parameter.

  When this change is made, any new instances that are launched are based on the new settings, but existing instances are not affected. To ensure that your Auto Scaling group is using the new settings, you can replace all instances in the group by starting an instance refresh or by using the maximum instance lifetime feature.

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

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

  ```
  {
    "AutoScalingGroupName":"my-asg",
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "Overrides":[
          {
            "InstanceType":"c6g.large",
            "LaunchTemplateSpecification": {
              "LaunchTemplateName": "my-launch-template-for-arm",
              "Version": "$Latest"
            }
          },
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          }
        ]
      }
    }
  }
  ```

To verify the launch templates for an Auto Scaling group

- Use the following **describe-auto-scaling-groups** command to verify and view the currently specified launch templates.

  ```
  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-0fb0e487336917fb2",
              "LaunchTemplateName":"my-launch-template-for-x86",
              "Version":"$Latest"
            },
            "Overrides":[
  ```

```
            {
              "InstanceType":"c6g.large",
              "LaunchTemplateSpecification": {
                "LaunchTemplateId": "lt-09d958b8fb2ba5bcc",
                  "LaunchTemplateName": "my-launch-template-for-arm",
                  "Version": "$Latest"
              }
            },
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":3,
      "Instances":[
        {
          "InstanceId":"i-07c63168522c0f620",
          "InstanceType":"c5.large",
          "AvailabilityZone":"us-west-2c",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchTemplate":{
            "LaunchTemplateId":"lt-0fb0e487336917fb2",
            "LaunchTemplateName":"my-launch-template-for-x86",
            "Version":"1"
          },
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-0b7ff78be9896a2c2",
          "InstanceType":"c5.large",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchTemplate":{
            "LaunchTemplateId":"lt-0fb0e487336917fb2",
            "LaunchTemplateName":"my-launch-template-for-x86",
            "Version":"1"
          },
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-0c682c2ceae918bc0",
          "InstanceType":"c6g.large",
          "AvailabilityZone":"us-west-2b",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchTemplate":{
            "LaunchTemplateId":"lt-09d958b8fb2ba5bcc",
            "LaunchTemplateName":"my-launch-template-for-arm",
            "Version":"1"
          },
      ...
    }
```

```
    ]
}
```

Configuring 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 Auto Scaling groups with multiple instance types and purchase options (p. 53).

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`)
  - 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": {
                "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": 0,
                    "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. 53).

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. 26).

- 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. 255).

To create an Auto Scaling group using a launch template (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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. Choose **Create an Auto Scaling group**.

4. 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**.

5. 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.

6. Under **Network**, for **VPC**, choose the VPC for the security groups that you specified in your launch template.

7. 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).

8. Choose **Next**.

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

9. (Optional) On the **Configure advanced options** page, configure the following options, and then choose **Next**:

   a. To register your Amazon EC2 instances with a load balancer, choose an existing load balancer or create a new one. For more information, see Elastic Load Balancing and Amazon EC2 Auto Scaling (p. 86). To create a new load balancer, follow the procedure in Create and attach a new Application Load Balancer or Network Load Balancer (p. 89).

   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.

10. (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. 112).

    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. 128).

    c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance scale-in protection (p. 191).

11. (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. 224).

12. (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. 82).

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

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 the Amazon EC2 launch wizard

You can create a launch template and an Auto Scaling group in a single procedure by using the Amazon EC2 launch wizard. This is useful if you want to create a new launch template 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 template.

To create a launch template 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 instances**.

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

4. In **Step 3: Configure Instance Details**, choose **Launch into Auto Scaling Group** next to **Number of instances**.

5. On the confirmation page, choose **Continue**.

6. On the **Create launch template** page, enter a name and description for the new launch template, and review the details of the template. If everything is satisfactory, choose **Create launch template**. Otherwise, update the settings of the launch template. For more information, see Creating your launch template (console) (p. 26).

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

8. On the **Choose launch template or configuration** page, the launch template that you created is already selected for you. Enter a name for the group, and then choose **Next**.

9. 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.

10. Under **Network**, specify a VPC and one or more subnets.

11. Choose **Next** twice.

12. (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. 112).

    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. 128).

13. You can optionally add notifications or tags. Or, you can choose **Skip to review**.

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

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. 25).

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. 46).

Prerequisites

Create a launch configuration. For more information, see Creating a launch configuration (p. 39).

To create an Auto Scaling group using a launch configuration (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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. Choose **Create an Auto Scaling group**.

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

5. 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**.

6. 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.

7. 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).

8. Choose **Next**.

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

9. (Optional) On the **Configure advanced options** page, configure the following options, and then choose **Next**:

   a. To register your Amazon EC2 instances with a load balancer, choose an existing load balancer or create a new one. For more information, see Elastic Load Balancing and Amazon EC2 Auto Scaling (p. 86). To create a new load balancer, follow the procedure in Create and attach a new Application Load Balancer or Network Load Balancer (p. 89).

   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.

10. (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. 112).

    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. 128).

    c. Under **Instance scale-in protection**, choose whether to enable instance scale-in protection. For more information, see Instance scale-in protection (p. 191).

11. (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. 224).

12. (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. 82).

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

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

Creating an Auto Scaling group might require that you configure and provision an Amazon EC2 instance first. For example, you might want to test that everything works the way that you intend. Multiple properties are required to create an EC2 instance, such as the AMI ID, instance type, key pair, and security group. All of this information is also required by Amazon EC2 Auto Scaling to launch instances on your behalf when there is a need to scale. This information is stored in a launch template or launch configuration.

You can create an Auto Scaling group using an existing EC2 instance in one of three ways.

- Create a launch template from an existing EC2 instance. Then use the launch template to create a new Auto Scaling group. For this procedure, see Creating a launch template from an existing instance (console) (p. 31).
- Use the console to create an Auto Scaling group from a running EC2 instance. When you do this, Amazon EC2 Auto Scaling creates a launch configuration for you and associates it with the Auto Scaling group. This method works well if you want to add the instance to the new Auto Scaling group where it can be managed by Amazon EC2 Auto Scaling. For more information, see Attach EC2 instances to your Auto Scaling group (p. 116).
- Specify the ID of an existing EC2 instance in the API call that creates the Auto Scaling group. This method is the subject of the following procedure.

When you specify an ID of an existing 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, including the AMI ID, instance type, key pair, and security group. The block devices come from the AMI that was used to launch the instance, not the instance.

Limitations and prerequisites

The following are limitations when using the following procedure to create 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 that was used to launch the instance. It does not include any block devices that were 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.

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

Amazon EC2 Auto Scaling User Guide
Create an Auto Scaling group
from an EC2 instance (AWS CLI)

Create an Auto Scaling group from an EC2 instance (AWS CLI)

The following examples show how to use the AWS CLI to create an Auto Scaling group from an EC2 instance.

To create an Auto Scaling group from an EC2 instance

- Use the following **create-auto-scaling-group** command to create an Auto Scaling group, `my-asg-from-instance`, from the EC2 instance `i-0e69cc3f05f825f4f`.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
  --instance-id i-0e69cc3f05f825f4f --min-size 1 --max-size 2 --desired-capacity 2
```

To verify that your Auto Scaling group has launched instances

- Use the following **describe-auto-scaling-groups** command to verify that the Auto Scaling group was created successfully.

```
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":[
    {
      "AutoScalingGroupName":"my-asg-from-instance",
      "AutoScalingGroupARN":"arn",
      "LaunchConfigurationName":"my-asg-from-instance",
      "MinSize":1,
      "MaxSize":2,
      "DesiredCapacity":2,
      "DefaultCooldown":300,
      "AvailabilityZones":[
        "us-west-2a"
      ],
      "LoadBalancerNames":[ ],
      "TargetGroupARNs":[ ],
      "HealthCheckType":"EC2",
      "HealthCheckGracePeriod":0,
      "Instances":[
        {
          "InstanceId":"i-06905f55584de02da",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-087b42219468eacde",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
```

Amazon EC2 Auto Scaling User Guide
Create an Auto Scaling group
from an EC2 instance (AWS CLI)

```
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        }
      ],
      "CreatedTime":"2020-10-28T02:39:22.152Z",
      "SuspendedProcesses":[ ],
      "VPCZoneIdentifier":"subnet-6bea5f06",
      "EnabledMetrics":[ ],
      "Tags":[ ],
      "TerminationPolicies":[
        "Default"
      ],
      "NewInstancesProtectedFromScaleIn":false,
      "ServiceLinkedRoleARN":"arn"
    }
  ]
}
```

To view the launch configuration

- Use the following **describe-launch-configurations** command to view the details of the launch configuration.

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

The following is example output:

```
{
  "LaunchConfigurations":[
    {
      "LaunchConfigurationName":"my-asg-from-instance",
      "LaunchConfigurationARN":"arn",
      "ImageId":"ami-0528a5175983e7f28",
      "KeyName":"my-key-pair-uswest2",
      "SecurityGroups":[
        "sg-05eaec502fcdadc2e"
      ],
      "ClassicLinkVPCSecurityGroups":[ ],
      "UserData":"",
      "InstanceType":"t2.micro",
      "KernelId":"",
      "RamdiskId":"",
      "BlockDeviceMappings":[ ],
      "InstanceMonitoring":{
        "Enabled":true
      },
      "CreatedTime":"2020-10-28T02:39:22.321Z",
      "EbsOptimized":false,
      "AssociatePublicIpAddress":true
    }
  ]
}
```

To terminate the instance

- If you no longer need the instance, you can terminate it. The following **terminate-instances** command terminates the instance `i-0e69cc3f05f825f4f`.

```
aws ec2 terminate-instances --instance-ids i-0e69cc3f05f825f4f
```

After you terminate an Amazon EC2 instance, you can't restart the instance. After termination, its data is gone and the volume can't be attached to any instance. To learn more about terminating instances, see Terminating an instance in the *Amazon EC2 User Guide for Linux Instances*.

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. 248). 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. 26).

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.

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 **Add 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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Tags**, **Edit**.
4. To modify existing tags, edit **Key** and **Value**.
5. 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.

6. When you have finished adding tags, choose **Update**.

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) (p. 85)
- Delete tags (AWS CLI) (p. 85)

Delete tags (console)

To delete a tag

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Tags**, **Edit**.
4. Choose **Remove** next to the tag.
5. Choose **Update**.

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

Elastic Load Balancing is used to automatically distribute your incoming application traffic across all the EC2 instances that you are running. You can use Elastic Load Balancing to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed.

To use Elastic Load Balancing with your Auto Scaling group, you attach the load balancer to your Auto Scaling group (p. 87) to register the group with the load balancer. Your load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group.

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

After attaching a load balancer to your Auto Scaling group, you can configure your Auto Scaling group to use Elastic Load Balancing metrics such as the Application Load Balancer request count per target (or other metrics) to scale the number of instances in the group as the demand on your instances changes.

You can also optionally enable Amazon EC2 Auto Scaling to replace instances in your Auto Scaling group based on health checks provided by Elastic Load Balancing. Otherwise, you can create a CloudWatch alarm that notifies you if the healthy host count of the target group is lower than allowed.

Limits

- The load balancer and its target group must be in the same Region where you create your Auto Scaling group.
- The target group must specify a target type of `instance`. You can't specify a target type of `ip` when using an Auto Scaling group.

Contents

Elastic Load Balancing types

Elastic Load Balancing provides four types of load balancers that can be used with your Auto Scaling group: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers.

There is a key difference in how the load balancer types are configured. With Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, instances are registered as targets with a target group, and you route traffic to the target group. With Classic Load Balancers, instances are registered directly with the load balancer.

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).

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.

Gateway Load Balancer

> Distributes traffic to a fleet of appliance instances, providing scale, availability, and simplicity for third-party virtual appliances, such as firewalls, intrusion detection and prevention systems, and other appliances. Gateway Load Balancers work with virtual appliances that support the GENEVE protocol. Additional technical integration is required, so make sure that you consult the user guide before choosing a Gateway Load Balancer.

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.

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

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

Attaching a load balancer to your Auto Scaling group

This topic describes how to attach an Elastic Load Balancing load balancer to your Auto Scaling group.

Amazon EC2 Auto Scaling integrates with Elastic Load Balancing to help you to insert an Application Load Balancer, Network Load Balancer, Classic Load Balancer, or Gateway Load Balancer in front of your Auto Scaling group. Note that Classic Load Balancers are the only type of load balancer available for EC2-Classic. To learn more about the different types of load balancers, see Elastic Load Balancing types (p. 86).

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, it enters the `Added` state. After at least one registered instance passes the health checks, it enters the `InService` state. When the load balancer is in 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.

> **Note**
> By default, connection draining is enabled for Application Load Balancers, Network Load Balancers, and Gateway Load Balancers, but must be enabled for Classic Load Balancers.

Contents

Prerequisites

If necessary, create a new launch template or launch configuration for your Auto Scaling group. Create a launch template or launch configuration with a security group that controls the traffic to and from the instances behind the load balancer. The recommended rules depend on the type of load balancer and the types of backends that the load balancer uses. For example, to route traffic to web servers, allow inbound HTTP access on port 80 from the load balancer.

Before you can deploy virtual appliances behind a Gateway Load Balancer, the launch template or launch configuration must meet the following requirements:

- The Amazon machine image (AMI) must specify the ID of an AMI that supports GENEVE using a Gateway Load Balancer.
- The security groups must allow UDP traffic on port 6081.

You can configure the health check settings for a specific Auto Scaling group at any time. For more information, see Adding Elastic Load Balancing health checks to an Auto Scaling group (p. 91). Before adding these health checks, make sure you have a security group for your launch template or launch configuration that allows access from the load balancer on the port on which you want Elastic Load Balancing to perform health checks.

Attach a load balancer (console)

Contents

Attach an existing load balancer

Use the following procedures to attach a load balancer when you create or update an Auto Scaling group.

Start by following the procedures in the Elastic Load Balancing documentation to create a new load balancer and target group. You can skip the step for registering your Amazon EC2 instances. Amazon EC2 Auto Scaling automatically takes care of registering instances. For more information, see Getting started with Elastic Load Balancing in the *Elastic Load Balancing User Guide*.

To attach an existing load balancer as you are creating a new Auto Scaling group

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Choose **Create Auto Scaling group**.
3. In Steps 1 and 2, choose the options as desired and proceed to **Step 3: Configure advanced options**.
4. For **Load balancing**, choose **Attach to an existing load balancer**.
5. Under **Attach to an existing load balancer**, do one of the following:

a. For Application Load Balancers, Network Load Balancers, and Gateway Load Balancers:

Choose **Choose from your load balancer target groups**, and then choose a target group in the **Existing load balancer target groups** field.

b. For Classic Load Balancers:

Choose **Choose from Classic Load Balancers**, and then choose your load balancer in the **Classic Load Balancers** field.

6. Proceed to create the Auto Scaling group. Your instances will be automatically registered to the load balancer after the Auto Scaling group has been created.

To attach an existing load balancer to an existing Auto Scaling group

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

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Load balancing**, **Edit**.
4. Under **Load balancing**, do one of the following:

   a. For **Application, Network or Gateway Load Balancer target groups**, select its check box and choose a target group.

   b. For **Classic Load Balancers**, select its check box and choose your load balancer.
5. Choose **Update**.

Create and attach a new Application Load Balancer or Network Load Balancer

To make it easier to set up a load-balanced application, you can use the following procedure to create and attach an Application Load Balancer or a Network Load Balancer as you create your Auto Scaling group.

Important
After creating your Auto Scaling group, if you need to create a listener with a secure protocol, such as HTTPS, or a UDP listener, you will need to use the Elastic Load Balancing console to create additional listeners. You can add another listener to an existing load balancer, as long as you use distinct ports.

To create and attach a new load balancer as you are creating a new 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. Choose **Create Auto Scaling group**.
4. In Steps 1 and 2, choose the options as desired and proceed to **Step 3: Configure advanced options**.
5. For **Load balancing**, choose **Attach to a new load balancer**.

   a. Under **Attach to a new load balancer**, for **Load balancer type**, choose whether to create an Application Load Balancer or Network Load Balancer.

   b. For **Load balancer name**, enter a name for the load balancer, or keep the default name.

   c. For **Load balancer scheme**, choose whether to create a public internet-facing load balancer, or keep the default for an internal load balancer.

      d.   For **Availability Zones and subnets**, for each Availability Zone that you chose to launch your EC2 instances into (which prepopulate from Step 2), select the public subnet in those Availability Zones.

      e.   For **Listeners and routing**, update the port number for your listener (if necessary), and under **Default routing**, choose **Create a target group**. Alternatively, you can choose an existing target group from the drop-down list.

      f.   If you chose **Create a target group** in the last step, for **New target group name**, enter a name for the target group, or keep the default name.

      g.   To add tags, choose **Add tag**, and provide a tag key and value for each tag.

6.   Proceed to create the Auto Scaling group. Your instances will be automatically registered to the load balancer after the Auto Scaling group has been created.

Attach a load balancer (AWS CLI)

To create an Auto Scaling group with an attached target group

Use the following **create-auto-scaling-group** command with the `--target-group-arns` option to create an Auto Scaling group with an attached target group. Specify the Amazon Resource Name (ARN) of a target group (where you register targets by instance ID) for an Application Load Balancer, Network Load Balancer, and Gateway Load Balancer.

```
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
```

To attach a target group to an existing Auto Scaling group

Use the following **attach-load-balancer-target-groups** command to attach a target group (where you register targets by instance ID) to your Auto Scaling group.

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-
targets/1234567890123456"
```

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 attach a Classic Load Balancer to an existing Auto Scaling group

Use the following **attach-load-balancers** command to attach the specified Classic Load Balancer to your Auto Scaling group.

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

Detach a load balancer

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

To detach a load balancer from a group

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

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
4. On the **Details** tab, choose **Load balancing**, **Edit**.
5. Under **Load balancing**, do one of the following:

   a. For **Application, Network or Gateway Load Balancer target groups**, choose the delete icon (X) next to the target group.

   b. For **Classic Load Balancers**, choose the delete icon (X) next to the load balancer.
6. Choose **Update**

To detach a target group (AWS CLI)

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 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-
targets/1234567890123456"
```

To detach a Classic Load Balancer (AWS CLI)

Use the following **detach-load-balancers** command to detach a Classic 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, it is marked unhealthy and will be terminated while Amazon EC2 Auto Scaling launches a new instance in replacement.

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

To ensure that your Auto Scaling group can determine an instance's health based on additional tests provided by the load balancer, you can configure the Auto Scaling group to use Elastic Load Balancing (ELB) health checks. The load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances and determines if an instance is unhealthy. If you configure the Auto Scaling group to use ELB health checks, it considers the instance unhealthy if it fails either the EC2 status checks or the ELB 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.

See the following topics:

- To configure health checks for your Application Load Balancer, see Health checks for your target groups in the *User Guide for Application Load Balancers*.
- To configure health checks for your Network Load Balancer, see Health checks for your target groups in the *User Guide for Network Load Balancers*.
- To configure health checks for your Gateway Load Balancer, see Health checks for your target groups in the *User Guide for Gateway Load Balancers*.
- To configure health checks for your Classic Load Balancer, see Configure health checks for your Classic Load Balancer in the *User Guide for Classic Load Balancers*.
- For more information about Amazon EC2 Auto Scaling health checks, see Health checks for Auto Scaling instances (p. 205).

The following procedures show how to add `ELB` health checks to your Auto Scaling group.

Contents

Adding health checks (console)

Use the following procedure to add an `ELB` health check to an Auto Scaling group with an attached load balancer.

To add health checks

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Health checks**, **Edit**.
4. For **Health check type**, select **Enable ELB health checks**.
5. For **Health check grace period**, enter the amount of time, in seconds, that Amazon EC2 Auto Scaling needs to wait before checking the health status of an instance. Frequently, new instances need to briefly warm up before they can pass a health check. To provide ample warm-up time, set the health check grace period of the group to match the expected startup time of your application.
6. Choose **Update**.
7. On the **Instance management** tab, under **Instances**, 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.

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.

Limits

- 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.
- To change which Availability Zones are enabled for your load balancer, you need to be aware of the following limitations:
  - When you enable an Availability Zone for your load balancer, you specify one subnet from that Availability Zone. Note that you can select at most one subnet per Availability Zone.
  - For internet-facing load balancers, the subnets that you specify for the load balancer must have at least 8 available IP addresses.
  - For Application Load Balancers, you must enable at least two Availability Zones.
  - For Network Load Balancers, you cannot disable the enabled Availability Zones, but you can enable additional ones.
  - For Gateway Load Balancers, you cannot change the Availability Zones or subnets that were added when the load balancer was created.

Contents

Add an Availability Zone (console)

Use the following procedure to expand your Auto Scaling group and load balancer to a subnet in an additional Availability Zone.

To add an Availability Zone

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. On the **Details** tab, choose **Network**, **Edit**.

4. In **Subnets**, choose the subnet corresponding to the Availability Zone.

5. Choose **Update**.

6. 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, under **LOAD BALANCING**, choose **Load Balancers**.

   b. Choose your load balancer.

   c. Do one of the following:

      - For Application Load Balancers and Network Load Balancers:

        1. On the **Description** tab, for **Availability Zones**, choose **Edit subnets**.

        2. On the **Edit subnets** page, for **Availability Zones**, select the check box for the Availability Zone to add, and then select one of the subnets for that Availability Zone.

      - For Classic Load Balancers in a VPC:

        1. On the **Instances** tab, choose **Edit Availability Zones**.

        2. 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**.

      - For Classic Load Balancers in EC2-Classic:

        1. On the **Instances** tab, choose **Edit Availability Zones**.

        2. On the **Add and Remove Availability Zones** page, choose the Availability Zone to add.

   d. Choose **Save**.

Add an Availability Zone (AWS CLI)

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

For an Auto Scaling group with an Application Load Balancer or Network 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 subnet-8360a9e7
   ```

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 or Network Load Balancer using the following **set-subnets** command.

   ```
   aws elbv2 set-subnets --load-balancer-arn my-lb-arn \
     --subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
   ```

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
```

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
```

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
```

Getting recommendations for an instance type from AWS Compute Optimizer

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 Compute Optimizer in the *AWS Compute Optimizer User Guide*.

Contents

-

Limitations

Compute Optimizer generates recommendations for instances in Auto Scaling groups that are configured to launch and run M, C, R, T, and X instance types. However, it does not generate recommendations for -g instance types powered by AWS Graviton2 processors (e.g., C6g), and for -n instance types that have higher network bandwidth performance (e.g., M5n).

The Auto Scaling groups must also be configured to run a single instance type (i.e., no mixed instance types), must not have a scaling policy attached to them, and have the same values for desired, minimum, and maximum capacity (i.e., an Auto Scaling group with a fixed number of instances). Compute Optimizer generates recommendations for instances in Auto Scaling groups that meet *all* of these configuration requirements.

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 the Limitations (p. 96) 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, update the launch template or update the Auto Scaling group to use a new launch configuration. Existing instances continue to use the previous

configuration. To update the existing instances, terminate them so that they are replaced by your Auto Scaling group, or allow automatic scaling to gradually replace older instances with newer instances based on your termination policies (p. 188).

> **Note**
> With the maximum instance lifetime and instance refresh features, you can also replace existing instances in your Auto Scaling group to launch new instances that use the new launch template or launch configuration. For more information, see Replacing Auto Scaling instances based on maximum instance lifetime (p. 97) and Replacing Auto Scaling instances based on an instance refresh (p. 100).

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 an SDK 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 86,400 seconds (1 day). 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. 191).

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
        }
    ]
}
```

Replacing Auto Scaling instances based on an instance refresh

When a configuration change requires replacing instances, and you have a large number of instances in your Auto Scaling group, it can be difficult to manually replace instances a few at a time. With an instance refresh, it's easier to update the instances in your Auto Scaling group.

During an instance refresh, Amazon EC2 Auto Scaling takes a set of instances out of service, terminates them, and then launches a set of instances with the new configuration. After that, instances are replaced on a rolling basis. In a rolling update, when a new instance launches, Amazon EC2 Auto Scaling waits until the instance passes a health check and completes warm-up before moving on to replace another instance. This process repeats until all instances are terminated and replaced.

This feature is helpful, for example, when you have a new launch template or launch configuration that specifies a new AMI or new user data. You just need to update your Auto Scaling group to specify the new launch template or launch configuration. Then start an instance refresh to immediately begin the process of replacing instances in the group.

Instance refreshes depend on health checks to determine whether your application is healthy enough to consider a replacement successful. For information about the various health checks that are used to determine an instance's health, see Health checks for Auto Scaling instances (p. 205).

Before starting an instance refresh, you can configure the minimum healthy percentage to control the level of disruption to your application. If your application doesn't pass health checks, the rolling update process waits for a period of up to 60 minutes after it reaches the minimum healthy threshold before it eventually fails. The intention is to give it time to recover in case of a temporary issue. If the rolling update process fails, any instances that were already replaced are not rolled back to their previous configuration. To fix a failed instance refresh, first resolve the underlying issue that caused the update to fail, and then initiate another instance refresh.

After determining that a newly launched instance is healthy, Amazon EC2 Auto Scaling does not immediately move on to the next replacement. It provides a window for each instance to warm up after launching, which you can configure. This can be helpful when you have configuration scripts that take time to run. To protect your application's availability, ensure that the instance warm-up period covers the expected startup time for your application, from when a new instance comes into service to when it can receive traffic.

The following are things to consider when starting an instance refresh, to help ensure that the group continues to perform as expected.

- While warming up, a newly launched instance is not counted toward the aggregated metrics of the Auto Scaling group.
- If you added scaling policies to the Auto Scaling group, the scaling activities run in parallel. If you set a long interval for the instance refresh warm-up period, it will take more time for newly launched instances to be reflected in the metrics. An adequate warm-up period therefore helps to prevent Amazon EC2 Auto Scaling from scaling on stale metric data.
- If you added a lifecycle hook to the Auto Scaling group, the warm-up period does not start until the lifecycle hook actions complete and the instance enters the `InService` state.

You can start or cancel an instance refresh using the AWS Management Console, the AWS CLI, or an SDK.

Start or cancel an instance refresh (console)

Before you begin, make sure that your Auto Scaling group is already associated with a new launch template or launch configuration.

To start an instance refresh

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. On the **Instance refresh** tab, in **Instance refreshes**, choose **Start instance refresh**.
4. In the **Start instance refresh** dialog box, do the following:

   - (Optional) For **Minimum healthy percentage**, keep the default value, 90, or specify a new value within the range of 0 percent to 100 percent. This is the amount of capacity in the Auto Scaling group that must remain healthy during an instance refresh to allow the operation to continue, expressed as a percentage of the group's desired capacity. Increasing this setting to 100 percent limits the rate of replacement to one instance at a time. In contrast, decreasing this setting to 0 percent has the effect of replacing all instances at the same time.
   - (Optional) For **Instance warmup**, the default is the value specified for the health check grace period for the group. You can change the value to ensure that it reflects your actual application startup time. The default value may not reflect very recent updates in the latest version of your application.
5. Choose **Start**.

To check the status of an instance refresh

1. On the **Instance refresh** tab, under **Instance refreshes**, you can determine the status of your request by looking at the **Status** column. The operation goes into Pending status while it is initializing. The status should then quickly change to InProgress. When all instances are updated, the status changes to Successful.
2. On the **Activity** tab, under **Activity history**, when the instance refresh starts, you see entries when instances are terminated and another set of entries when instances are launched. In the **Description** column, you can find the instance ID.
3. On the **Instance management** tab, under **Instances**, you can verify that your instances launched successfully. Initially, your instances are in the Pending state. 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 cancel an instance refresh

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.
3. On the **Instance refresh** tab, in **Instance refreshes**, choose **Cancel instance refresh**.
4. When prompted for confirmation, choose **Confirm**.

Start or cancel an instance refresh (AWS CLI)

To start an instance refresh

To start an instance refresh from the AWS CLI, use the start-instance-refresh command. You can specify any preferences that you want to change in a JSON configuration file. When you reference the configuration file, provide the file's path and name as shown in the following example.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of `config.json`.

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 400,
      "MinHealthyPercentage": 50
    }
}
```

Alternatively, you can start the instance refresh without the optional preferences by running the following command. If preferences are not provided, the default values are used for `InstanceWarmup` and `MinHealthyPercentage`.

```
aws autoscaling start-instance-refresh --auto-scaling-group-name my-asg
```

Example output.

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

To check the status of an instance refresh

View the instance refreshes for an Auto Scaling group by using the following describe-instance-refreshes command.

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

Example output.

```
{
    "InstanceRefreshes": [
        {
            "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
            "AutoScalingGroupName": "my-asg",
            "Status": "InProgress",
            "StartTime": "2020-06-02T18:11:27Z",
            "PercentageComplete": 0,
            "InstancesToUpdate": 5
        },
        {
            "InstanceRefreshId": "dd7728d0-5bc4-4575-96a3-1b2c52bf8bb1",
            "AutoScalingGroupName": "my-asg",
            "Status": "Successful",
            "StartTime": "2020-06-02T16:43:19Z",
            "EndTime": "2020-06-02T16:53:37Z",
            "PercentageComplete": 100,
            "InstancesToUpdate": 0
        }
    ]
}
```

To cancel an instance refresh

When you cancel an instance refresh using the cancel-instance-refresh command from the AWS CLI, specify the name of the Auto Scaling group as shown in the following example.

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

Example output.

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

Limitations

- **Total duration**: The maximum amount of time that an instance refresh can continue to actively replace instances is 14 days.
- **Instances not replaced**: If an instance is on standby or protected from scale in, it cannot be replaced. If Amazon EC2 Auto Scaling encounters an instance that it cannot replace, it will continue to replace other instances.
- **One-hour timeout**: When an instance refresh is unable to continue making replacements because there are instances on standby or protected from scale in, it keeps retrying for an hour and provides a status message to help you resolve the issue. If the problem persists after an hour, the operation fails.
- **No rollback**: You can cancel an instance refresh at any time, but any instances that have already been replaced are not rolled back to their previous configuration.

Adding checkpoints to an instance refresh

When using an instance refresh, you have the option to replace the entire Auto Scaling group in one continuous operation. However, you might prefer to replace the group in phases, so that you can perform verifications on your instances as you go. To do a phased replacement, you add checkpoints, which are points in time where the instance refresh pauses. Using checkpoints gives you greater control over how you choose to update your Auto Scaling group, and it helps you to ensure that your application will function in a reliable, predictable manner.

To enable checkpoints for an instance refresh, add the following refresh preferences to your configuration file:

- `CheckpointPercentages`: Specifies threshold values for the percentage of instances to be replaced. These threshold values provide the checkpoints. When the percentage of instances that are replaced and warmed up reaches one of the specified thresholds, the operation waits for a specified period of time. You specify the number of seconds to wait in `CheckpointDelay`. When the specified period of time has passed, the instance refresh continues until it reaches the next checkpoint (if applicable).
- `CheckpointDelay`: The amount of time, in seconds, to wait after a checkpoint is reached before continuing. Choose a time period that allows you enough time to perform your validations.

The percentage of the Auto Scaling group that needs to be successfully replaced is indicated by the last value shown in the `CheckpointPercentages` array. The operation doesn't transition to `Successful` until this percentage of the group is replaced successfully, and each instance is warmed up and ready to start serving traffic again.

Amazon EC2 Auto Scaling emits events for each checkpoint. If you add an EventBridge rule to send the events to a target such as Amazon SNS, you can be notified when you can run the required validations. For more information, see Creating EventBridge rules for instance refresh events (p. 234).

Setting instance refresh checkpoint preferences (console)

You can configure checkpoints in the preferences for an instance refresh.

To start an instance refresh that uses checkpoints

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. On the **Instance refresh** tab, in **Instance refreshes**, choose **Start instance refresh**.
4. In the **Start instance refresh** dialog box, specify the applicable values for **Minimum healthy percentage** and **Instance warmup**.
5. Select the **Enable instance refresh with checkpoints** check box.

   This displays a box where you can define the percentage threshold for the first checkpoint.
6. To set a percentage for the first checkpoint, enter a number from 1 to 100 in the number field in **Proceed until ____ % of the group is refreshed**.
7. To add another checkpoint, choose **Add checkpoint** and then define the percentage for the next checkpoint.
8. To specify how long Amazon EC2 Auto Scaling waits after a checkpoint is reached, update the fields in **Wait for 1 hour between checkpoints**. The time unit can be hours, minutes, or seconds.
9. Choose **Start**.

Setting instance refresh checkpoint preferences (AWS CLI)

To create multiple checkpoints

To create multiple checkpoints, use the following example start-instance-refresh command. This example configures an instance refresh that refreshes 1 percent of the Auto Scaling group initially, waits 10 minutes, then refreshes the next 19 percent, waits 10 minutes, and then refreshes the rest of the group before succeeding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of `config.json`.

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 400,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20,100],
      "CheckpointDelay": 600
    }
}
```

To create a single checkpoint

To create a single checkpoint, use the following example start-instance-refresh command. This example configures an instance refresh that refreshes 20 percent of the Auto Scaling group initially, waits 10 minutes, and then refreshes the rest of the group before concluding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of `config.json`.

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 400,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [20,100],
      "CheckpointDelay": 600
    }
}
```

To only partially refresh the Auto Scaling group

To replace a portion of your Auto Scaling group and then stop completely, use the following example start-instance-refresh command. This example configures an instance refresh that refreshes 1 percent of the Auto Scaling group initially, waits 10 minutes, and then refreshes the next 19 percent before concluding the operation.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

Contents of `config.json`.

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 400,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20],
      "CheckpointDelay": 600
    }
}
```

Key points about checkpoints

The following are key points to note about using checkpoints:

- A checkpoint is reached when the number of instances replaced reaches the percentage threshold that is defined for the checkpoint. The percentage of instances replaced could be at or above, but not below, the percentage threshold.
- After a checkpoint is reached, the overall percentage complete does not immediately display the latest status until the instances finish warming up.

  For example, assume your Auto Scaling group has 10 instances and makes the following replacements. Your checkpoint percentages are [20,50] with a checkpoint delay of 15 minutes and a minimum healthy percentage of 80%.
  - 0:00: 2 old instances are replaced.
  - 0:10: 2 new instances finish warming up.
  - 0:25: 2 old instances are replaced. Only two instances are replaced to maintain the minimum healthy percentage.
  - 0:35: 2 new instances finish warming up.
  - 0:35: 1 old instance is replaced.
  - 0:45: 1 new instance finishes warming up.

  At 0:35, the operation will stop launching new instances even though the percentage complete won't yet accurately reflect the number of replacements complete at 50% until 10 minutes later when the new instance completes its warm-up period.

- Because checkpoints are based on percentages, the number of instances to replace to reach a checkpoint changes with the size of the group. This means that when a scale-out activity occurs and the size of the group increases, an in-progress operation could reach a checkpoint again. If that happens, Amazon EC2 Auto Scaling sends another notification and repeats the wait time between checkpoints before continuing.
- It's possible to skip a checkpoint under certain circumstances. For example, suppose your Auto Scaling group has 2 instances and your checkpoint percentages are `[10,40,100]`. After the first instance is replaced, Amazon EC2 Auto Scaling calculates that 50% of the group has been replaced. Because 50% is higher than the first two checkpoints, it skips the first checkpoint (`10`) and sends a notification for the second checkpoint (`40`).
- Cancelling the operation stops any further replacements from being made. If you cancel the operation or it fails before reaching the last checkpoint, any instances that were already replaced are not rolled back to their previous configuration.
- In the case of a partial refresh, when you rerun the operation, Amazon EC2 Auto Scaling doesn't restart from the point of the last checkpoint and stop when only the old instances are replaced. However, it will target old instances for replacement first before targeting new instances.

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group and choose **Delete**.
3. 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, under **INSTANCES**, 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 and target groups

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, under **LOAD BALANCING**, 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, under **LOAD BALANCING**, 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. 113).

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. 114).

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. 153).

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. 123).

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 desired capacity must be greater than or equal to the minimum size of the group and less than or equal to the maximum size of the group.

An Auto Scaling group will start by launching as many instances as are specified for desired capacity. If there are no scaling policies or scheduled actions attached to the Auto Scaling group, Amazon EC2 Auto Scaling maintains the desired amount of instances, performing periodic health checks on the instances in the group. Unhealthy instances will be terminated and replaced with new ones.

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 auto 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 auto scaling and have the group run at a fixed size, either temporarily or permanently.

To manage these 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. On the **Auto Scaling groups** page, select the check box next to the Auto Scaling group whose settings you want to manage.
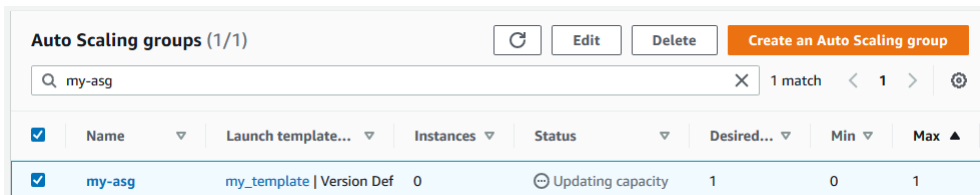
   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
4. In the lower pane, in the **Details** tab, view or change the current settings for minimum, maximum, and desired capacity.

Above the **Details** pane contains overview information about the Auto Scaling group, including the current number of instances in the group, the minimum, maximum, and desired capacity, and a status column. If the Auto Scaling group uses instance weighting, then the information includes the number of capacity units contributed to the desired capacity. To add or remove columns from the list, choose the settings icon at the top of the page. Then, for **Auto Scaling groups attributes**, turn each column on or off, and choose **Confirm** changes.

To verify the size of your Auto Scaling group after making changes

The **Instances** column shows the number of instances that are currently running. While an instance is being launched or terminated, the **Status** column displays a status of *Updating capacity*, as shown in the following image.

Wait for a few minutes, and then refresh the view to see the latest status. After a scaling activity completes, notice that the **Instances** column shows a new value.

You can also view the number of instances and the status of the instances that are currently running from the **Instance management** tab under **Instances**.

Maintaining a fixed number of instances in your Auto Scaling group

Amazon EC2 Auto Scaling lets you set various parameters to have an Auto Scaling group maintain a fixed size. You can then choose whether to manually add or remove Amazon EC2 instances from the group to deal with traffic changes to your application.

If a fixed number of instances is needed, this can be achieved by setting the same value for minimum, maximum, and desired capacity. After you have created your Auto Scaling group, the group starts by launching enough instances to meet its 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.

Amazon EC2 Auto Scaling monitors the health of each Amazon EC2 instance that it launches. 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. If any instance terminates unexpectedly, Amazon EC2 Auto Scaling detects the termination and launches a replacement instance. This capability enables you to maintain a fixed, desired number of EC2 instances automatically. For more information about health check replacements, see Health checks for Auto Scaling instances (p. 205).

Amazon EC2 Auto Scaling also lets you adjust the desired capacity to update the number of instances that it 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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. On the **Details** tab, choose **Group details**, **Edit**.

4. 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**.

5. When you are finished, choose **Update**.

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. 268).

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. 150).

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, under **INSTANCES**, 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, for **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 check box next to the new Auto Scaling group, choose the **Edit** button that is above the list of groups, change the settings as needed, and then choose **Update**.

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, on the **Details** tab, increase the maximum capacity.
3. On the navigation pane, under **INSTANCES**, 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, for **Auto Scaling Group**, select the Auto Scaling group, 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. 116).
- 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. 199).

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Instance management** tab, in **Instances**, select an instance and choose **Actions**, **Detach**.
4. 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"
        }
    ]
}
```

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. 69) 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 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 Application Load Balancer target group at 1000 for your Auto Scaling group.

Depending on your application needs, you might find that one of these popular scaling metrics works best for you when using target tracking, or you might find that a combination of these metrics or a different metric meets your needs better.

Contents

Considerations

Before you create a target tracking scaling policy for your Auto Scaling group, you should understand the following characteristics and behaviors of target tracking scaling policies:

- 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.
- 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. For more information, see Multiple scaling policies (p. 125).

- 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 managed by AWS and deleted automatically when no longer needed.

Choosing metrics

In a target tracking scaling policy, you can use predefined or customized metrics.

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 or your own metrics in CloudWatch by specifying a customized metric. You must use the AWS CLI or an SDK to create a target tracking policy with 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 Amazon 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 Scaling based on Amazon SQS (p. 140).

- A target tracking scaling policy does not scale in your Auto Scaling group when the specified metric has no 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.

- To use the `ALBRequestCountPerTarget` metric, you must specify the `ResourceLabel` parameter to identify the target group that is associated with the metric.

Monitoring Amazon EC2 metrics

To ensure a faster response to changes in the metric value, we recommend that you scale on metrics with a 1-minute frequency. Scaling on metrics with a 5-minute frequency can result in slower response times and scaling on stale metric data.

To get this level of data for Amazon EC2 metrics, you must specifically enable detailed monitoring. By default, Amazon EC2 instances are enabled for basic monitoring, which means metric data for instances is available at 5-minute frequency. For more information, see Configuring monitoring for Auto Scaling instances (p. 221).

Instance warm-up

When you create a target tracking scaling 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.

While scaling out, we do not consider instances that are warming up as part of the current capacity of the group so that the metrics are not affected by higher resource usage at startup. 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 (console)

You can choose to configure a target tracking scaling policy on an Auto Scaling group as you create it or after the Auto Scaling group is created.

To create an Auto Scaling group with a target tracking scaling policy

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Choose **Create Auto Scaling group**.
3. In Steps 1, 2, and 3, choose the options as desired and proceed to **Step 4: Configure group size and scaling policies**.
4. Under **Group size**, specify the range that you want to scale between by updating the minimum capacity and maximum capacity. These two settings allow your Auto Scaling group to scale dynamically. Amazon EC2 Auto Scaling scales your group in the range of values specified by the minimum capacity and maximum capacity.
5. Under **Scaling policies**, choose **Target tracking scaling policy**.
6. To define a policy, do the following:

   a. Specify a name for the policy.

   b. For **Metric type**, choose a metric.

      If you chose **Application Load Balancer request count per target**, choose a target group in **Target group**.

   c. Specify a **Target value** for the metric.

   d. (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.

e. (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.

7. Proceed to create the Auto Scaling group. Your scaling policy will be created after the Auto Scaling group has been created.

To create a target tracking scaling policy for an existing Auto Scaling group

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. Verify that the minimum capacity and maximum capacity 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, on the **Details** tab, change the current settings for minimum and maximum capacity.

4. On the **Automatic scaling** tab, in **Scaling policies**, choose **Add 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. For **Metric type**, choose a metric. You can choose only one metric type. To use more than one metric, create multiple policies.

        If you chose **Application Load Balancer request count per target**, choose a target group in **Target group**.

    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 (p. 129)
- Step 2: Create a target tracking scaling policy (p. 130)

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 by calling **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

After you have created the Auto Scaling group, you can create a target tracking scaling policy that instructs Amazon EC2 Auto Scaling 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 target tracking configuration that keeps the average CPU utilization at 40 percent. Save this configuration in a file named `config.json`.

```
{
  "TargetValue": 40.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
}
```

For more information, see PredefinedMetricSpecification in the *Amazon EC2 Auto Scaling API Reference*.

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"
    }
}
```

For more information, see CustomizedMetricSpecification in the *Amazon EC2 Auto Scaling API Reference*.

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
```

If successful, this command returns the ARNs and names of the two CloudWatch alarms created on your behalf.

```
{
    "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-scaling-
policy",
```

```
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-asg-
AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-asg-
AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

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. 125). 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.

Contents

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 (p. 150) 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 for step scaling

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
20	null	30

Example: Step adjustments for scale-in policy

Lower bound	Upper bound	Adjustment
-10	0	0

Lower bound	Upper bound	Adjustment
-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 an SDK, 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 instance weighting (p. 69) 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 largest 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 a CloudWatch alarm (console)

You can use the following procedure to create the CloudWatch alarms that Amazon EC2 Auto Scaling uses to determine when to scale your Auto Scaling group. Each CloudWatch alarm watches a single metric and sends messages to Amazon EC2 Auto Scaling when the metric breaches the alarm threshold.

To create a CloudWatch alarm that monitors CPU utilization

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region. From the navigation bar, select the region where your Auto Scaling group resides.
3. In the navigation pane, choose **Alarms** and then choose **Create alarm**.
4. Choose **Select metric**.
5. On the **All metrics** tab, choose **EC2**, **By Auto Scaling Group**, and enter the Auto Scaling group's name in the search field. Then, select `CPUUtilization` and choose **Select metric**. The **Specify metric and conditions** page appears, showing a graph and other information about the metric.
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. Under **Conditions**, do the following:

   - For **Threshold type**, choose **Static**.
   - For **Whenever `CPUUtilization` is**, specify whether you want the value of the metric to be greater than, greater than or equal to, less than, or less than or equal to the threshold to trigger the alarm. Then, under **than**, enter the threshold value that you want to trigger the alarm.

8. Under **Additional configuration**, do the following:

   - For **Datapoints to alarm**, enter the number of data points (evaluation periods) during which the metric value must meet the threshold conditions to trigger the alarm. For example, two consecutive periods of 5 minutes would take 10 minutes to trigger the alarm.
   - For **Missing data treatment**, choose **Treat missing data as bad (breaching threshold)**. For more information, see Configuring how CloudWatch alarms treat missing data in the *Amazon CloudWatch User Guide*.

9. Choose **Next**.
10. (Optional) Under **Notification**, you can choose or create the Amazon SNS topic you want to use to receive notifications. Otherwise, you can remove the notification now and add one later as needed.
11. Choose **Next**.
12. Enter a name (for example, `Step-Scaling-AlarmHigh-AddCapacity`) and, optionally, a description for the alarm, and then choose **Next**.
13. Choose **Create alarm**.

Example: CloudWatch alarm that triggers whenever CPU breaches the 80 percent threshold
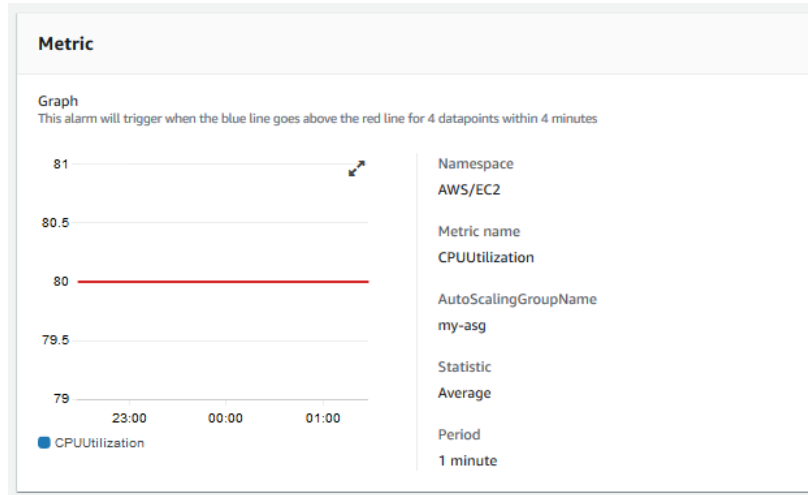
Create step scaling policies (console)

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.

While you are configuring your scaling policies, you can create the alarms at the same time. Alternatively, you can use alarms that you created from the CloudWatch console, as described in the previous section.

To create a step scaling policy for scale out

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. 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, on the **Details** tab, change the current settings for minimum and maximum capacity.

4. On the **Automatic scaling** tab, in **Scaling policies**, choose **Add policy**.

5. To define a policy for scale out (increase capacity), do the following:

   a. For **Policy type**, choose **Step scaling**.

   b. Specify a name for the policy.

   c. For **CloudWatch alarm**, choose your alarm. If you haven't already created an alarm, choose **Create a CloudWatch alarm** and complete Step 4 (p. 135) through Step 13 (p. 135) to create an alarm that monitors CPU utilization. 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**.

b. Specify a name for the policy.

c. For **CloudWatch alarm**, choose your alarm. If you haven't already created an alarm, choose **Create a CloudWatch alarm** and complete Step 4 (p. 135) through Step 13 (p. 135) to create an alarm that monitors CPU utilization. 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**.

Create scaling policies and CloudWatch alarms (AWS CLI)

Use the AWS CLI as follows to configure step or simple 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 by calling **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.

Step scaling policies

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
```

Record the policy's Amazon Resource Name (ARN). You need it to create a CloudWatch alarm for the policy.

```
{
    "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
```

Record the policy's Amazon Resource Name (ARN). You need it to create a CloudWatch alarm for the policy.

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-
cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy
}
```

Simple scaling policies

Alternatively, you can create simple scaling policies by using the following CLI commands instead of the preceding CLI commands. 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
```

Record the policy's Amazon Resource Name (ARN). You need it to create a CloudWatch alarm for the policy.

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 one instance.

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

Record the policy's Amazon Resource Name (ARN). You need it to create a CloudWatch alarm for the policy.

Step 3: Create CloudWatch alarms

In step 2, you created scaling policies that provided instructions to the Auto Scaling group about how to scale 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. 127).

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.
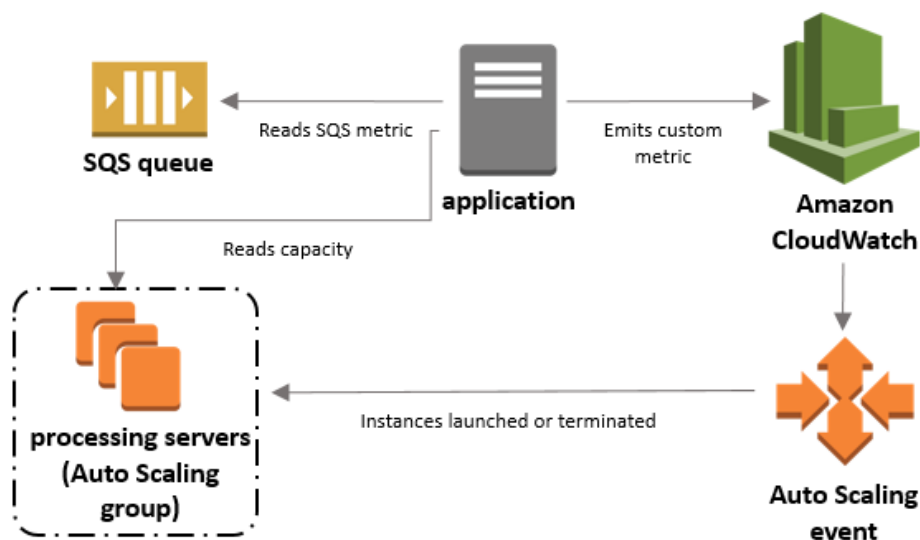
Limitations and prerequisites

To use this configuration, you need to be aware of the following limitations:

- You must use the AWS CLI or an SDK to publish your custom metric to CloudWatch. You can then monitor your metric with the AWS Management Console.
- After publishing your custom metric, you must use the AWS CLI or an SDK 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. For example, to get metric data that reflects the present use of the queue, you use the SQS **get-queue-attributes** command. 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. 224).

To view the scaling activities for an Auto Scaling group (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances.
4. On the **Instance management** tab, under **Instances**, the **Lifecycle** column contains the state of your instances. It takes a short time for an instance to launch. After the instance starts, its lifecycle state changes to `InService`.

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

To view the scaling activities for an 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
\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

For information about the types of errors that you may encounter and how to handle them, see Troubleshooting Amazon EC2 Auto Scaling (p. 268).

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 (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. 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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Automatic scaling** tab, in **Scaling policies**, select a scaling policy, and then choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.
5. (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 get the scaling policies for an Auto Scaling group (AWS CLI)

Before you delete a scaling policy, use the following **describe-policies** command to see what scaling policies were created for the Auto Scaling group. You can use the output when deleting the policy and the CloudWatch alarms.

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

You can filter the results by the type of scaling policy using the `--query` parameter. This syntax for `query` works on Linux or macOS. On Windows, change the single quotes to double quotes.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

The following is example output.

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "PolicyName": "cpu40-target-tracking-scaling-policy",
        "PolicyARN": "PolicyARN",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-
asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
```

```
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:region:account-id:alarm:TargetTracking-my-
asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
                "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
            }
        ],
        "TargetTrackingConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ASGAverageCPUUtilization"
            },
            "TargetValue": 40.0,
            "DisableScaleIn": false
        },
        "Enabled": true
    }
]
```

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 cpu40-target-tracking-scaling-policy
```

To delete your CloudWatch alarm (AWS CLI)

For step 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 SDKs.

The following examples show how you can create scaling policies for Amazon EC2 Auto Scaling 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. 125) and Step and simple scaling policies (p. 131).

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/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
  },
  "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 --min-adjustment-magnitude 2
```

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 -1 \
  --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

A scaling cooldown helps you prevent your Auto Scaling group from launching or terminating additional instances before the effects of previous activities are visible.

When you use simple scaling, after the Auto Scaling group scales using a simple scaling policy, it waits for a cooldown period to complete before any further scaling activities initiated by simple scaling policies can start. An adequate cooldown period helps to prevent the initiation of an additional 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. For more information about simple scaling, see Step and simple scaling policies (p. 131).

> **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 (p. 125) over either simple scaling or step scaling.

During a cooldown period, when a scheduled action starts at the scheduled time, or when scaling activities due to target tracking or step scaling policies start, they can trigger a scaling activity immediately without waiting for the cooldown period to expire. If an instance becomes unhealthy, Amazon EC2 Auto Scaling also does not wait for the cooldown period to complete before replacing the unhealthy instance.

When you manually scale your Auto Scaling group, the default is not to wait for the cooldown period to complete, but you can override this behavior and honor the cooldown period when you call the API.

Contents

- Default cooldown period (p. 151)
- Scaling-specific cooldown period (p. 151)

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. You can configure the length of time based on your instance startup time or other application needs.
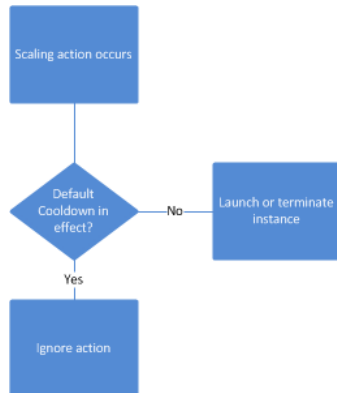
When you use the AWS Management Console to update an Auto Scaling group, or when you use the AWS CLI or an SDK to create or update an Auto Scaling group, you can set the optional default cooldown parameter. If a value for the default cooldown period is not provided, its default value is 300 seconds.

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. 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. 131).

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.
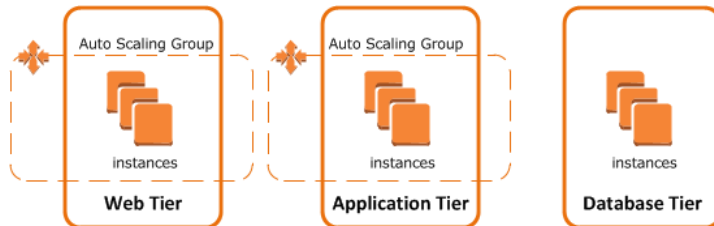
To help ensure that the group for the application tier has the appropriate number of EC2 instances, you create a simple scaling policy to scale out whenever the value for the CloudWatch metric associated with the scaling policy exceeds a specified threshold for consecutive specified periods. When the 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.
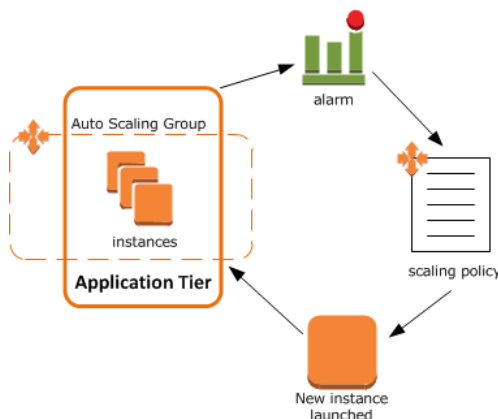
However, with a cooldown period in place, the Auto Scaling group launches an instance and then blocks scaling activities due to simple scaling policies until the specified time elapses. (The default is 300 seconds.) This gives newly launched instances time to start handling application traffic. After 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. 158).

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.

Scheduled scaling for Amazon EC2 Auto Scaling

Scheduled scaling helps you to set up your own scaling schedule according to predictable load changes. 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 configure a schedule for Amazon EC2 Auto Scaling to increase capacity on Wednesday and decrease capacity on Friday.

To use scheduled scaling, you create *scheduled actions*. Scheduled actions are performed automatically as a function of date and time. When you create a scheduled action, you specify when the scaling activity should occur and the new desired, minimum, and maximum sizes for the scaling action. You can create scheduled actions that scale one time only or that scale on a recurring schedule.

Contents

Considerations

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

- A scheduled action sets the desired, minimum, and maximum sizes to what is specified by the scheduled action at the date and time specified. The request can optionally include only one of these sizes. For example, you can create a scheduled action with only the desired capacity specified. In some cases, however, you must include the minimum and maximum sizes to ensure that the new desired capacity is not outside of these limits.
- By default, the times that you set are in Coordinated Universal Time (UTC). When specifying a recurring schedule with a cron expression using the AWS CLI or an SDK, you can change the time zone to correspond to your local time zone or a time zone for another part of your network. When you specify a time zone that observes Daylight Saving Time (DST), the recurring action automatically adjusts for Daylight Saving Time.
- If you specify a recurring schedule, you can specify a date and time for the start time, the end time, or both.
    - Choose your start and end times carefully. The start time and end time must be set in UTC.
    - If you specify a start time, Amazon EC2 Auto Scaling performs the action at this time, and then performs the action based on the specified recurrence.
    - If you specify an end time, the action stops repeating after this time. A scheduled action does not persist in your account once it has reached its end time.
- If needed, you can disable and re-enable scheduled scaling at any time. For more information, see Suspending and resuming scaling processes (p. 199).
- The order of execution for scheduled actions is guaranteed within the same group, but not for scheduled actions across groups.
- 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.

    Warning
    Currently, the time zone feature is available only if you use the AWS CLI or an SDK, and is not available from the console. If you specify a recurring schedule with a time zone, the cron expression uses the specified time zone, which you can check by describing the scheduled action from the CLI or an SDK. If you use the console to update a scheduled action with a recurring schedule, this update action deletes the time zone (if one was set), which defaults the cron expression to UTC.

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. On the **Automatic scaling** tab, in **Scheduled actions**, choose **Create scheduled action**.

4. Enter a **Name** for the scheduled action.

5. For **Desired capacity**, **Min**, **Max**, choose the new desired size of the group and the new minimum and maximum capacity.

6. For **Recurrence**, choose one of the available options.

   - If you want to scale only once, choose **Once**.
   - If you want to scale on a recurring basis, choose how often Amazon EC2 Auto Scaling should run the scheduled action.
     - If you choose an option that begins with **Every**, the cron expression is created for you.
     - If you choose **Cron**, enter a cron expression that specifies when to perform the action, in UTC. The cron expression format consists of five fields separated by white space. For more information about this format, see Crontab. For examples of cron expressions, see https://crontab.guru/examples.html.

7. Define a **Start time**, in UTC.

   - If you chose **Once** as the recurrence, the start time defines the date and time for the schedule action to run.
   - If you chose an option that runs on a recurring basis, the start time defines when the first scheduled action in the recurring series runs.

8. (Optional) For recurrent actions, you can specify a value for **End time**, in UTC.

9. Choose **Create**.

   The console displays the scheduled actions for the Auto Scaling group.

To create a scheduled action for an Auto Scaling 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. Select your Auto Scaling group.

4. On the **Scheduled Actions** tab, choose **Create Scheduled Action**.

5. On the **Create Scheduled Action** page, do the following:

   - Specify the size of the group using at least one of the following values: **Min**, **Max**, or **Desired Capacity**.
   - For **Recurrence**, choose one of the available options.
     - If you want to scale only once, choose **Once**.
     - If you want to scale on a recurring basis, choose how often Amazon EC2 Auto Scaling should run the scheduled action.
       - If you choose an option that begins with **Every**, the cron expression is created for you.
       - If you choose **Cron**, enter a cron expression that specifies when to perform the action, in UTC. The cron expression format consists of five fields separated by white space. For more information about this format, see Crontab. For examples of cron expressions, see https://crontab.guru/examples.html.

- Define a **Start time**.
  - If you chose **Once** as the recurrence, the start time defines the date and time for the schedule action to run.
  - If you chose an option that runs on a recurring basis, the start time defines when the first scheduled action in the recurring series runs.
  - (Optional) If you specified a recurring schedule, you can specify a value for **End Time**.
6. Choose **Create**.

   The console displays the scheduled actions for the Auto Scaling group.

Update a scheduled action

After creating a scheduled action, you can update any of its settings except the name and the time zone. When you edit a scheduled action, all times are in UTC.

To update a scheduled action (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action.
4. Choose **Actions**, **Edit**.
5. Make the needed changes and choose **Save changes**.

Delete a scheduled action

When you no longer need a scheduled action, you can delete it.

To delete a scheduled action (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select your Auto Scaling group.
3. On the **Automatic scaling** tab, in **Scheduled actions**, select a scheduled action.
4. Choose **Actions**, **Delete**.
5. 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.

Create a scheduled action that occurs only once

To automatically scale your Auto Scaling group one time only, at a specified date and time, use the `--start-time "YYYY-MM-DDThh:mm:ssZ"` option.

Example: To scale out one time only

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` (8:00 AM UTC on March 31, 2021), if the group currently has fewer than 3 instances, it 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 "2021-03-31T08:00:00Z" --desired-capacity 3
```

Example: To scale in one time only

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` (4:00 PM UTC on March 31, 2021), if the group currently has more than 1 instance, it 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 "2021-03-31T16:00:00Z" --desired-capacity 1
```

Create a scheduled action that runs on a recurring schedule

To schedule scaling on a recurring schedule, use the `--recurrence "cron expression"` option. The supported cron expression 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. For examples of cron expressions, see https://crontab.guru/examples.html.

The following is an example of a scheduled action that specifies a cron expression.

On the specified schedule (every day at 9:00 AM UTC), if the group currently has fewer than 3 instances, it scales out to 3 instances. If the group currently has more than 3 instances, it scales in to 3 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
  --auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

Create a recurring scheduled action that specifies a time zone

Scheduled actions are set to the UTC time zone by default. To specify a different time zone, include the `--time-zone` option and specify the canonical name for the IANA time zone (`America/New_York`, for example). For more information, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.

The following is an example that uses the `--time-zone` option when creating a recurring scheduled action to scale capacity.

On the specified schedule (every Monday through Friday at 6:00 PM local time), if the group currently has fewer than 2 instances, it scales out to 2 instances. If the group currently has more than 2 instances, it scales in to 2 instances.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
  --auto-scaling-group-name my-asg --recurrence "0 18 * * 1-5" --time-zone "America/New_York" \
```

```
    --desired-capacity 2
```

Describe scheduled actions

To describe the scheduled actions for an Auto Scaling group, use the following **describe-scheduled-actions** command.

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

If successful, this command returns output similar to the following.

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
            "StartTime": "2020-12-01T00:30:00Z",
            "Time": "2020-12-01T00:30:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4
        }
    ]
}
```

Delete a scheduled action

To delete a scheduled action, use the following **delete-scheduled-action** command.

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action
```

Limitations

- The names of scheduled actions must be unique per Auto Scaling group.
- A scheduled action must have a unique time value. If you attempt to schedule an activity at a time when another scaling activity is already scheduled, the call is rejected and returns an error indicating that a scheduled action with this scheduled start time already exists.
- You can create a maximum of 125 scheduled actions per Auto Scaling group.
- You can only specify a time zone for a cron expression.

Amazon EC2 Auto Scaling lifecycle hooks

Amazon EC2 Auto Scaling offers the ability to add lifecycle hooks to your Auto Scaling groups. These hooks enable an Auto Scaling group to be aware of events in the Auto Scaling instance lifecycle, and then perform a custom action when the corresponding lifecycle event occurs. A lifecycle hook provides a specified amount of time (one hour by default) to complete the lifecycle action before the instance transitions to the next state.

As an example of using lifecycle hooks with Auto Scaling instances:

- When a scale-out event occurs, your newly launched instance completes its startup sequence and transitions to a wait state. While the instance is in a wait state, it runs a script to download and install the needed software packages for your application, making sure that your instance is fully ready before it starts receiving traffic. When the script is finished installing software, it sends the **complete-lifecycle-action** command to continue.
- When a scale-in event occurs, a lifecycle hook pauses the instance before it is terminated and sends you a notification using Amazon EventBridge. While the instance is in the wait state, you can invoke an AWS Lambda function or connect to the instance to download logs or other data before the instance is fully terminated.

A popular use of lifecycle hooks is to control when instances are registered with Elastic Load Balancing. By adding a launch lifecycle hook to your Auto Scaling group, you can ensure that the applications on the instances are ready to accept traffic before they are registered to the load balancer at the end of the lifecycle hook.

Contents

> **Note**
> Amazon EC2 Auto Scaling provides its own lifecycle to help with the deployment of Auto Scaling groups. This lifecycle differs from that of other EC2 instances. For more information, see Amazon EC2 Auto Scaling instance lifecycle (p. 7).

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 a scale-out event and begins launching an instance.
2. The lifecycle hook puts the instance into a wait state (`Pending:Wait`) and then performs a custom action.

   The instance remains in a wait state either until you complete the lifecycle action using the complete-lifecycle-action CLI command or the CompleteLifecycleAction action, or until the timeout period ends. By default, the instance remains in a wait state for one hour, and then the Auto Scaling group continues the launch process (`Pending:Proceed`). If you need more time, you can restart the timeout period by recording a heartbeat. If you complete the lifecycle action before the timeout period expires, the period ends and the Auto Scaling group continues the launch process.
3. The instance enters the `InService` state and the health check grace period starts. If the Auto Scaling group is associated with an Elastic Load Balancing load balancer, the instance is registered with the load balancer, and the load balancer starts checking its health. After the health check grace period ends, Amazon EC2 Auto Scaling begins checking the health state of the instance.

4. The Auto Scaling group responds to a scale-in event and begins terminating an instance. If the Auto Scaling group is being used with Elastic Load Balancing, the terminating instance is first deregistered from the load balancer. After it's deregistered, the instance stops accepting new connections and waits for existing connections to drain before continuing.

5. The lifecycle hook puts the instance into a wait state (`Terminating:Wait`) and then performs a custom action.

   The instance remains in a wait state either until you complete the lifecycle action, or until the timeout period ends (one hour by default). After you complete the lifecycle hook or the timeout period expires, the instance transitions to the next state (`Terminating:Proceed`).

6. The instance is terminated.

The following illustration shows the transitions between Auto Scaling instance states in this process.
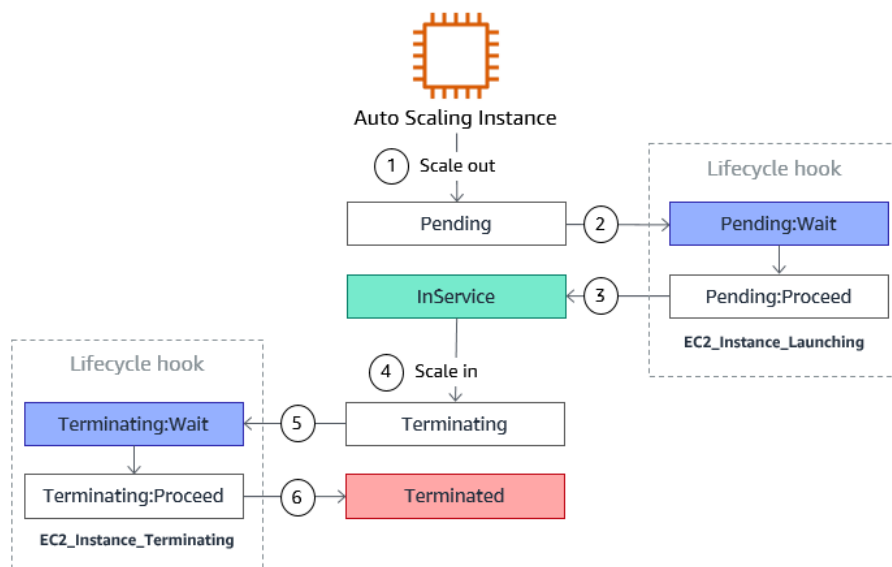
Considerations and limitations

When using lifecycle hooks, keep in mind the following key considerations and limitations:

- You can configure a launch lifecycle hook to abandon the launch if an unexpected failure occurs, in which case Amazon EC2 Auto Scaling automatically terminates and replaces the instance.
- Amazon EC2 Auto Scaling limits the rate at which it allows instances to launch if the lifecycle hooks are failing consistently.
- 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, which can happen at any time with a two-minute interruption notice. For more information, see Spot Instance interruptions in the *Amazon EC2 User Guide for Linux Instances*.
- When scaling out, Amazon EC2 Auto Scaling doesn't count a new instance towards the aggregated CloudWatch metrics of the Auto Scaling group until after the launch lifecycle hook finishes (and the scaling policy's warm-up time completes). On scale in, the terminating instance continues to count as part of the group's aggregated metrics until after the termination lifecycle hook finishes.
- When an Auto Scaling group launches or terminates an instance due to simple scaling policies, 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 actions finish, 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. 150).

- There is a quota that specifies the maximum number of lifecycle hooks that you can create per Auto Scaling group. For more information, see Amazon EC2 Auto Scaling service quotas (p. 10).
- For examples of the use of lifecycle hooks, see the following blog posts: Building a Backup System for Scaled Instances using AWS Lambda and Amazon EC2 Run Command and Run code before terminating an EC2 Auto Scaling instance.

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 how long the timeout period lasts 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.

Receiving lifecycle hook notifications

You can configure various EventBridge rules to invoke a Lambda function or send a notification when an instance enters a wait state and Amazon EC2 Auto Scaling submits an event for a lifecycle action to EventBridge. The event contains information about the instance that is launching or terminating, and a token that you can use to control the lifecycle action. For an introductory tutorial-style guide for creating lifecycle hooks, see Tutorial: Configure a lifecycle hook that invokes a Lambda function (p. 168). For examples of the events that are emitted when a lifecycle action occurs, see Auto Scaling events (p. 228).

Alternatively, if you have a user data or cloud-init script that configures your instances after they launch, you do not need to configure a notification. The script can control the lifecycle action using the ID of the instance on which it runs. 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*. You can have the script signal the lifecycle hook when the configuration script is complete, to allow the instance to proceed to the next state.

> **Note**
> The Amazon EC2 Auto Scaling console does not provide the option to define an Amazon SNS or Amazon SQS notification target for the lifecycle hook. These lifecycle hooks must be added using either the AWS CLI or an AWS SDK. For more information, see Configuring notifications (p. 162).

GitHub repository

You can visit our GitHub repository to download templates and scripts for lifecycle hooks.

Lifecycle hook availability

The following table lists the lifecycle hooks available for various scenarios.

Event	Instance launch or termination[1]	Maximum Instance Lifetime: Replacement instances	Instance Refresh: Replacement instances	Capacity Rebalancing: Replacement instances	Warm Pools: Instances entering and leaving the warm pool
Instance launching	✓	✓	✓	✓	✓
Instance terminating	✓	✓	✓	✓	✓

[1] Applies to instances launched or terminated when the group is created or deleted, when the group scales automatically, or when you manually adjust your group's desired capacity. Does not apply when you attach or detach instances, move instances in and out of standby mode, or delete the group with the force delete option.

Configuring a notification target for a lifecycle hook

You can add lifecycle hooks to an Auto Scaling group to perform custom actions whenever an instance enters a wait state. You can configure notification targets to perform these actions using a variety of AWS services depending on your preferred development approach.

The first approach uses Amazon EventBridge to invoke a Lambda function that performs the action you want. The second approach involves creating an Amazon Simple Notification Service (Amazon SNS) topic to which notifications are published. Clients can subscribe to the SNS topic and receive published messages using a supported protocol. The last approach involves using Amazon Simple Queue Service (Amazon SQS), a messaging system that requires worker nodes to poll a queue.

As a best practice, we recommend that you use EventBridge. The notifications sent to Amazon SNS and Amazon SQS contain the same information as the notifications that Amazon EC2 Auto Scaling sends to EventBridge. Before EventBridge, the standard practice was to send a notification to SNS or SQS and integrate another service with SNS or SQS to perform programmatic actions. Today, EventBridge gives you more options for which services you can target and makes it easier to handle events using serverless architecture.

The following procedures cover how to set up your notification target.

Remember, if you have user data scripts or cloud-init directives that configure your instances when they launch, you do not need to receive notification when the lifecycle action occurs.

Contents

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.

Route notifications to Lambda using EventBridge

You can invoke a Lambda function when a lifecycle action occurs using EventBridge. For information about the EventBridge events that are emitted when a lifecycle action occurs, see Auto Scaling events (p. 228). For an introductory tutorial-style guide that shows you how to create a simple Lambda function that listens for launch events and writes them out to a CloudWatch Logs log, see Tutorial: Configure a lifecycle hook that invokes a Lambda function (p. 168).

To use a Lambda function as the target of an EventBridge rule

1. Create a Lambda function by using the Lambda console and note its Amazon Resource Name (ARN). For example, `arn:aws:lambda:region:123456789012:function:my-function`. You need the ARN to create an EventBridge target.

   For more information, see Getting started with Lambda in the *AWS Lambda Developer Guide*.

2. Create an EventBridge rule that matches the lifecycle action using the following put-rule command.

   > **Note**
   > When you use the AWS Management Console to create an event rule, the console automatically adds the IAM permissions necessary to grant EventBridge permission to call your Lambda function. If you are creating an event rule using the AWS CLI, you need to grant this permission explicitly. For help creating an event rule using the console, see Tutorial: Configure a lifecycle hook that invokes a Lambda function (p. 168).

   ```
   aws events put-rule --name my-rule --event-pattern file://pattern.json --state ENABLED
   ```

   The following example shows the `pattern.json` for an instance launch lifecycle action.

   ```
   {
     "source": [ "aws.autoscaling" ],
     "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
   }
   ```

   The following example shows the `pattern.json` for an instance terminate lifecycle action.

   ```
   {
     "source": [ "aws.autoscaling" ],
     "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ]
   }
   ```

3. Create a target that invokes your Lambda function when the lifecycle action occurs, using the following put-targets command.

   ```
   aws events put-targets --rule my-rule --targets
    Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
   ```

4. Grant the rule permission to invoke your Lambda function using the following add-permission command. This command trusts the EventBridge service principal (`events.amazonaws.com`) and scopes permissions to the specified rule.

   ```
   aws lambda add-permission --function-name my-function --statement-id my-unique-id \
     --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
    arn:aws:events:region:123456789012:rule/my-rule
   ```

5. After you have followed these instructions, continue on to Adding lifecycle hooks (p. 166) as a next step.

Receive notifications using Amazon SNS

You can use Amazon SNS to set up a notification target (an SNS topic) to receive notifications when a lifecycle action occurs. Amazon SNS then sends the notifications to the subscribed recipients. Until the subscription is confirmed, no notifications published to the topic are sent to the recipients.

To set up notifications using Amazon SNS

1. Create an Amazon SNS topic by using either the Amazon SNS console or the following create-topic command. Ensure that the topic is in the same Region as the Auto Scaling group that you're using. For more information, see Getting started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

   ```
   aws sns create-topic --name my-sns-topic
   ```

2. Note the topic Amazon Resource Name (ARN), for example, `arn:aws:sns:region:123456789012:my-sns-topic`. You need it to create the lifecycle hook.

3. Create an IAM service role to give Amazon EC2 Auto Scaling access to your Amazon SNS notification target.

   To give Amazon EC2 Auto Scaling access to your SNS topic

   a. Open the IAM console at https://console.aws.amazon.com/iam/.

   b. On the navigation pane, choose **Roles**, **Create 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 need it to create the lifecycle hook.

4. After you have followed these instructions, continue on to Add lifecycle hooks (AWS CLI) (p. 167) as a next step.

Receive notifications using Amazon SQS

You can use Amazon SQS to set up a notification target to receive messages when a lifecycle action occurs. Worker nodes must then poll an SQS queue to act on these notifications.

Important
FIFO queues are not compatible with lifecycle hooks.

To set up notifications using Amazon SQS

1. Create an Amazon SQS queue by using the Amazon SQS console. Ensure that the queue is in the same Region as the Auto Scaling group that you're using. For more information, see Getting started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*.

2. Note the queue ARN, for example, `arn:aws:sqs:region:123456789012:my-sqs-queue`. You need it to create the lifecycle hook.

3. Create an IAM service role to give Amazon EC2 Auto Scaling access to your Amazon SQS notification target.

   To give Amazon EC2 Auto Scaling access to your SQS queue

   a. Open the IAM console at https://console.aws.amazon.com/iam/.

   b. On the navigation pane, choose **Roles**, **Create 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 need it to create the lifecycle hook.

4. After you have followed these instructions, continue on to as a next step.

Notification message example for Amazon SNS and Amazon SQS

While the instance is in a wait state, a message is published to the Amazon SNS or Amazon SQS notification target. The message includes the following information:

- `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.
- `NotificationMetadata` — The notification metadata.

The following is a notification message example.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
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: hook message metadata
```

Test notification message example

When you first add a lifecycle hook, a test notification message is published to the notification target. The following is a test notification message example.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

> **Note**
> For examples of the events delivered from Amazon EC2 Auto Scaling to EventBridge, see Auto
> Scaling events (p. 228).

Adding lifecycle hooks

After your notification target is set up and ready to use, add the lifecycle hook so that the Event
Notification can be used to perform your custom action when the corresponding lifecycle event occurs.

There are two types of lifecycle hooks that can be implemented: launch lifecycle hooks and termination
lifecycle hooks. Use a launch lifecycle hook to prepare instances for use or to delay instances from
registering behind the load balancer before their configuration has been applied completely. Use a
termination lifecycle hook to prepare running instances to be shut down.

Pay attention to the following settings when creating your lifecycle hook:

- **Heartbeat timeout**: This setting specifies how much time must pass before the hook times out. The
  range is from `30` to `7200` seconds. The default value is one hour (`3600` seconds). During the timeout
  period, you can, for example, install applications or download logs or other data.
- **Default result**: This setting defines the action to take when the lifecycle hook timeout elapses or when
  an unexpected failure occurs. You can choose to either *abandon* (default) 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 Amazon EC2 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.

Contents
- Add lifecycle hooks (console) (p. 166)
- Add lifecycle hooks (AWS CLI) (p. 167)

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 for the lifecycle hook to be
completed before abandoning or continuing.

To add a lifecycle hook

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.

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

   A split pane opens up in the bottom part of the **Auto Scaling groups** page.

3. On the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.

4. 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.

   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.

   e. (Optional) For **Notification metadata**, specify additional information that you want to include when Amazon EC2 Auto Scaling sends a message to the notification target.

5. 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 a notification target for a lifecycle hook (p. 162).

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"
```

Completing a lifecycle action

When an Auto Scaling group responds to a lifecycle event, it puts the instance in a wait state and sends an Event Notification. You can perform a custom action while the instance is in a wait state.

Completing a lifecycle action (AWS CLI)

The following procedure is for the command line interface and is not supported in the console. Information that must be replaced, such as the instance ID or the name of an Auto Scaling group, are shown in italics.

Amazon EC2 Auto Scaling User Guide
Tutorial: Configure a lifecycle hook
that invokes a Lambda function

To complete a lifecycle action

1. 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 that 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
```

2. 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
```

Tutorial: Configure a lifecycle hook that invokes a Lambda function

In this exercise, you create an EventBridge rule that includes a filter pattern that when matched, invokes an AWS Lambda function as the rule target. We provide the filter pattern and sample function code to use.

If everything is configured correctly, at the end of this tutorial, the Lambda function performs a custom action when instances launch. The custom action simply logs the event in the CloudWatch Logs log stream associated with the Lambda function.

The Lambda function also performs a callback to let the lifecycle of the instance proceed if this action is successful, but lets the instance abandon the launch and terminate if the action fails.

Contents

Amazon EC2 Auto Scaling User Guide
Tutorial: Configure a lifecycle hook
that invokes a Lambda function

Prerequisites

Before you begin this tutorial, create an Auto Scaling group, if you don't have one already. To create an Auto Scaling group, open the Auto Scaling groups page in the Amazon EC2 console and choose **Create Auto Scaling group**.

Note that all of the following procedures are for the new console.

Step 1: Create an IAM role with permissions to complete lifecycle hooks

Before you create a Lambda function, you must first create an execution role and a permissions policy to allow Lambda to complete lifecycle hooks.

To create the policy

1. Open the Policies page of the IAM console, and then choose **Create policy**.
2. Choose the **JSON** tab.
3. In the **Policy Document** box, paste the following policy document into the box, replacing the text in *italics* with your account number and the name of your Auto Scaling group.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-asg"
    }
  ]
}
```

4. Choose **Next:Tags**, and then **Next:Review**.
5. For **Name**, enter `LogAutoScalingEvent-policy`. Choose **Create policy**.

When you finish creating the policy, you can create a role that uses it.

To create the role

1. On the navigation pane, choose **Roles**, **Create role**.
2. Under **Choose a use case**, choose **Lambda** from the list, and then choose **Next:Permissions**.
3. Under **Attach permissions policies**, choose **LogAutoScalingEvent-policy** and **AWSLambdaBasicExecutionRole**.
4. Choose **Next:Tags**, and then **Next:Review**.
5. On the **Review** page, for **Name**, enter `LogAutoScalingEvent-role` and choose **Create role**.

Amazon EC2 Auto Scaling User Guide
Tutorial: Configure a lifecycle hook
that invokes a Lambda function

Step 2: Create a Lambda function

Create a Lambda function to serve as the target for events. The sample Lambda function, written in Node.js, is invoked by EventBridge when a matching event is emitted by Amazon EC2 Auto Scaling.

To create a Lambda function

1. Open the Functions page on the Lambda console.
2. Choose **Create function**, **Author from scratch**.
3. Under **Basic information**, for **Function name**, enter `LogAutoScalingEvent`.
4. Choose **Change default execution role**, and then for **Execution role**, choose **Use an existing role**.
5. For **Existing role**, choose **LogAutoScalingEvent-role**.
6. Leave the other default values.
7. Choose **Create function**. You are returned to the function's code and configuration.
8. With your `LogAutoScalingEvent` function still open in the console, under **Function code**, in the editor, copy the following sample code into the file named index.js.

```
var aws = require("aws-sdk");
exports.handler = (event, context, callback) => {
    console.log('LogAutoScalingEvent');
    console.log('Received event:', JSON.stringify(event, null, 2));
    var autoscaling = new aws.AutoScaling({region: event.region});
    var eventDetail = event.detail;
    var params = {
        AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
        LifecycleActionResult: 'CONTINUE', /* required */
        LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
        InstanceId: eventDetail['InstanceId'],
        LifecycleActionToken: eventDetail['LifecycleActionToken']
    };
    var response;
    autoscaling.completeLifecycleAction(params, function(err, data) {
        if (err) {
            console.log(err, err.stack); // an error occurred
            response = {
                statusCode: 500,
                body: JSON.stringify('ERROR'),
            };
        } else {
            console.log(data);              // successful response
            response = {
                statusCode: 200,
                body: JSON.stringify('SUCCESS'),
            };
        }
    });
    return response;
};
```

This code simply logs the event so that, at the end of this tutorial, you can see an event appear in the CloudWatch Logs log stream that's associated with this Lambda function.

9. Choose **Deploy**.

Step 3: Create an EventBridge rule

Create an EventBridge rule to run your Lambda function.

Amazon EC2 Auto Scaling User Guide
Tutorial: Configure a lifecycle hook
that invokes a Lambda function

To create a rule using the console

1. Open the EventBridge console.
2. On the navigation pane, choose **Rules**, **Create rule**.
3. For **Name**, enter **LogAutoScalingEvent-rule**.
4. For **Define pattern**, choose **Event Pattern**.
5. For **Event matching pattern**, choose **Custom pattern**.
6. Rules use event patterns to select events and route them to targets. Copy the following pattern into the **Event pattern** box.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

7. To save the event pattern, choose **Save**.
8. For **Select event bus**, choose **AWS default event bus**.
9. For **Target**, choose **Lambda function**.
10. For **Function**, select **LogAutoScalingEvent**. Choose **Create**.

Step 4: Add a lifecycle hook

In this section, you add a lifecycle hook so that Lambda runs your function on instances at launch.

To add a lifecycle hook

1. Open the Auto Scaling groups page in the Amazon EC2 console.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. In the lower pane, on the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.
4. To define a lifecycle hook, do the following:

   a. For **Lifecycle hook name**, enter **LogAutoScalingEvent-hook**.
   b. For **Lifecycle transition**, choose **Instance launch**.
   c. For **Heartbeat timeout**, enter **300** for the number of seconds to wait for a callback from your Lambda function.
   d. For **Default result**, choose **ABANDON**. This means that the Auto Scaling group will terminate a new instance if the hook times out without receiving a callback from your Lambda function.
   e. (Optional) Leave **Notification metadata** empty. The event data that we pass to EventBridge contains all of the necessary information to invoke the Lambda function.
5. Choose **Create**.

Step 5: Test and verify the event

To test the event, update the Auto Scaling group by increasing the desired capacity of the Auto Scaling group by 1. Your Lambda function is invoked within a few seconds after increasing the desired capacity.

To increase the size of the Auto Scaling group

1. Open the Auto Scaling groups page in the Amazon EC2 console.

Amazon EC2 Auto Scaling User Guide
Tutorial: Configure a lifecycle hook
that invokes a Lambda function

2. Select the check box next to your Auto Scaling group to view details in a lower pane and still see the top rows of the upper pane.

3. In the lower pane, on the **Details** tab, choose **Group details**, **Edit**.

4. For **Desired capacity**, increase the current value by 1.

5. Choose **Update**. While the instance is being launched, the **Status** column in the upper pane displays a status of *Updating capacity*.

After increasing the desired capacity, you can verify that your Lambda function was invoked.

To view the output from your Lambda function

1. Open the Log groups page of the CloudWatch console.

2. Select the name of the log group for your Lambda function (`/aws/lambda/LogAutoScalingEvent`).

3. Select the name of the log stream to view the data provided by the function for the lifecycle action.

Next, you can verify that your instance has successfully launched from the description of scaling activities.

To view the scaling activity

1. Return to the **Auto Scaling groups** page and select your group.

2. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched an instance.

   - If the action was successful, the scaling activity will have a status of "Successful".

   - If it failed, after waiting a few minutes, you will see a scaling activity with a status of "Cancelled" and a status message of "Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result".

To decrease the size of the Auto Scaling group

If you do not need the additional instance that you launched for this test, you can open the **Details** tab and decrease **Desired capacity** by 1.

Step 6: Next steps

Now that you have completed this tutorial, you can try creating a termination lifecycle hook. If instances in the Auto Scaling group terminate, an event is sent to EventBridge. For information about the event that is emitted when an instance terminates, see EC2 Instance-terminate Lifecycle Action (p. 230).

Step 7: Clean up

If you are done working with the resources that you created just for this tutorial, use the following steps to delete them.

To delete the lifecycle hook

1. Open the Auto Scaling groups page in the Amazon EC2 console.

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

3. On the **Instance management** tab, in **Lifecycle hooks**, choose the lifecycle hook (`LogAutoScalingEvent-hook`).

4. Choose **Actions**, **Delete**.

5. Choose **Delete** again to confirm.

To delete the Amazon EventBridge rule

1. Open the Rules page in the Amazon EventBridge console.

2. Under **Event bus**, choose the event bus that is associated with the rule (`Default`).

3. Choose the rule (`LogAutoScalingEvent-rule`).

4. Choose **Actions**, **Delete**.

5. Choose **Delete** again to confirm.

If you are done working with the example function, delete it. You can also delete the log group that stores the function's logs, and the execution role and permissions policy that you created.

To delete a Lambda function

1. Open the Functions page on the Lambda console.

2. Choose the function (`LogAutoScalingEvent`).

3. Choose **Actions**, **Delete**.

4. In the **Delete function** dialog box, choose **Delete**.

To delete the log group

1. Open the Log groups page of the CloudWatch console.

2. Select the function's log group (`/aws/lambda/LogAutoScalingEvent`).

3. Choose **Actions**, **Delete log group(s)**.

4. In the **Delete log group(s)** dialog box, choose **Delete**.

To delete the execution role

1. Open the Roles page of the IAM console.

2. Select the function's role (`LogAutoScalingEvent-role`).

3. Choose **Delete role**.

4. In the **Delete role** dialog box, choose **Yes, delete**.

To delete the IAM policy

1. Open the Policies page of the IAM console.

2. Select the policy that you created (`LogAutoScalingEvent-policy`).

3. Choose **Delete policy**.

4. In the **Delete policy** dialog box, choose **Yes, delete**.

Warm pools for Amazon EC2 Auto Scaling

A warm pool gives you the ability to decrease latency for your applications that have exceptionally long boot times, for example, because instances need to write massive amounts of data to disk. With warm pools, you no longer have to over-provision your Auto Scaling groups to manage latency in order to improve application performance.

This section shows how to add a warm pool to your Auto Scaling group. A warm pool is a pool of pre-initialized EC2 instances that sits alongside the Auto Scaling group. Whenever your application needs to scale out, the Auto Scaling group can draw on the warm pool to meet its new desired capacity. The goal of a warm pool is to ensure that instances are ready to quickly start serving application traffic, accelerating the response to a scale-out event. This is known as a *warm start*.

> **Important**
> Creating a warm pool when it's not required can lead to unnecessary costs. If your first boot time does not cause noticeable latency issues for your application, there probably isn't a need for you to use a warm pool.

You have the option of keeping instances in the warm pool in one of two states: `Stopped` or `Running`. Keeping instances in a `Stopped` state is an effective way to minimize costs. With stopped instances, you pay only for the volumes that you use and the Elastic IP addresses that are not assigned to a running instance. But you don't pay for the stopped instances themselves. You pay for the instances only when they are running.

The size of the warm pool is calculated as the difference between two numbers: the Auto Scaling group's maximum capacity and its desired capacity. For example, if the desired capacity of your Auto Scaling group is 6 and the maximum capacity is 10, the size of your warm pool will be 4 when you first set up the warm pool and the pool is initializing. The exception is if you specify a value for **Max prepared capacity**, in which case the size of the warm pool is calculated as the difference between the **Max prepared capacity** and the desired capacity. For example, if the desired capacity of your Auto Scaling group is 6, if the maximum capacity is 10, and if the **Max prepared capacity** is 8, the size of your warm pool will be 2 when you first set up the warm pool and the pool is initializing.

As instances leave the warm pool, they count toward the desired capacity of the group. There are two scenarios in which instances in the warm pool are replenished: when the group scales in and its desired capacity decreases, or when the group scales out and the minimum size of the warm pool is reached.

Lifecycle hooks control when new instances transition to and from the warm pool. These hooks help you make sure that your instances are fully configured for your application before they are added to the warm pool at the end of the lifecycle hook. On the next scale-out event, the instances are then ready to be added to the Auto Scaling group as soon as possible. Lifecycle hooks can also help you to ensure that instances are ready to serve traffic as they are leaving the warm pool (for example, by populating their cache). For more information, see Warm pool instance lifecycle (p. 177).

You can use the AWS Management Console, the AWS CLI, or an AWS SDK to add a warm pool to an Auto Scaling group.

Contents

Before you begin

Decide how you will use lifecycle hooks to prepare the instances for use. For example, you can configure lifecycle hooks to delay instances from being added to the warm pool before they complete their user

data script. Another way to use lifecycle hooks is to run code in a Lambda function triggered by lifecycle events. For more information, see the following blog post Scaling your applications faster with EC2 Auto Scaling Warm Pools.

Consider adjusting the warm-up time in your target tracking or step scaling policy to the minimum amount of time necessary. As instances leave the warm pool, Amazon EC2 Auto Scaling doesn't count them towards the aggregated CloudWatch metrics of the Auto Scaling group until after the lifecycle hook finishes and the warm-up time completes. If the warm-up time delays an `InService` instance from being included in the aggregated group metrics, you might find that your scaling policy scales out more aggressively than necessary. For more information, see Dynamic scaling for Amazon EC2 Auto Scaling (p. 123).

Add a warm pool (console)

The following procedure demonstrates the steps for adding a warm pool to an Auto Scaling group.

To add a warm pool

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page.
3. Choose the **Instance management** tab.
4. Under **Warm pool**, choose **Create warm pool**.
5. To configure a warm pool, do the following:

   a. For **Warm pool instance state**, choose which state you want to transition your instances to when they enter the warm pool. The default is `Stopped`.

   b. For **Minimum warm pool size**, enter the minimum number of instances to maintain in the warm pool.

   c. (Optional) For **Max prepared capacity**, choose **Define a set number of instances** if you want to control how much capacity is available in the warm pool.

   If you choose **Define a set number of instances**, you must enter the maximum instance count. This value represents the maximum number of instances that are allowed to be in the warm pool and the Auto Scaling group at the same time. If you enter a value that is less than the group's desired capacity, and you chose not to specify a value for **Minimum warm pool size**, the capacity of the warm pool will be 0.

   > **Note**
   > Keeping the default **Equal to the Auto Scaling group's maximum capacity** option helps you maintain a warm pool that is sized to match the difference between the Auto Scaling group's maximum capacity and its desired capacity. To make it easier to manage your warm pool, we recommend that you use the default so that you do not need to remember to adjust your warm pool settings in order to control the size of the warm pool.

6. Choose **Create**.

Add a warm pool (AWS CLI)

The following examples show how to add a warm pool with the AWS CLI put-warm-pool command. Each example shows how you can specify the command in each of the example scenarios.

Contents

Example 1: Keeping instances in the `Stopped` state

The following warm pool configuration is for a warm pool that keeps instances in a `Stopped` state. While instances are stopped, you don't have to pay any EC2 instance usage fees, though you are charged for any other resources that are attached to your stopped instances such as EBS volumes and Elastic IP addresses.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped
```

Example 2: Keeping instances in the `Running` state

The following example configuration is for a warm pool that keeps instances in a `Running` state instead of a `Stopped` state. You pay for instances while they are running. The costs incurred are based on the instance type.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Running
```

Example 3: Specifying the minimum number of instances in the warm pool

The following warm pool configuration specifies a minimum of 4 instances in the warm pool, so that this number of instances is always ready to handle traffic spikes.

```
aws aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 4
```

Example 4: Defining a warm pool size separate from the maximum group size

Generally, because you have a good understanding of how much above your desired capacity to set your maximum capacity, there is no need to define an additional maximum size. Amazon EC2 Auto Scaling simply creates a warm pool that dynamically resizes based on where your maximum capacity is set.

However, in cases where you want to have control over the group's maximum capacity and not have it impact the size of the warm pool, you can use the `--max-group-prepared-capacity` option. You might need to use this option when working with very large Auto Scaling groups to manage the cost benefits of having a warm pool. For example, an Auto Scaling group with 1000 instances, a maximum capacity of 1500 (to provide extra capacity for emergency traffic spikes), and a warm pool of 100 instances might achieve your objectives better than a warm pool of 500 instances.

The following warm pool configuration is for a warm pool that defines its size separately from the maximum group size. Suppose the Auto Scaling group has a desired capacity of 800. The size of the warm pool will be 100 when you run this command and the pool is initializing.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900
```

To maintain a minimum number of instances in the warm pool, include the `--min-size` option with the command, as follows.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

Example 5: Defining an absolute warm pool size

If you set the values for the `--max-group-prepared-capacity` and `--min-size` options to the same value, the warm pool will have an absolute size. The following warm pool configuration is for a warm pool that maintains a constant warm pool size of 10 instances.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

Limitations

- You cannot add a warm pool to Auto Scaling groups that have a mixed instances policy or that launch Spot Instances.
- You can put an instance in a `Stopped` state only if it has an Amazon EBS volume as its root device. Instances that use instance stores for the root device cannot be stopped.
- If your warm pool is depleted when there is a scale-out event, instances will launch directly into the Auto Scaling group (a *cold start*). You could also experience cold starts if an Availability Zone is out of capacity.
- Warm pools currently can't be used with ECS, EKS, and self-managed Kubernetes.

Warm pool instance lifecycle

Instances in the warm pool maintain their own independent lifecycle to help you create the appropriate lifecycle actions for each transition. This lifecycle is designed to facilitate the use of Event Notifications to invoke actions in a target service (for example, a Lambda function).

Note that the APIs that you use to create and manage lifecycle hooks have not changed (only the instance lifecycle has changed).

An Amazon EC2 instance transitions through different states from the moment it launches through to its termination. You can create lifecycle hooks to act on these event states, when an instance has transitioned from one state to another one.

The following diagram shows the transition between each state:
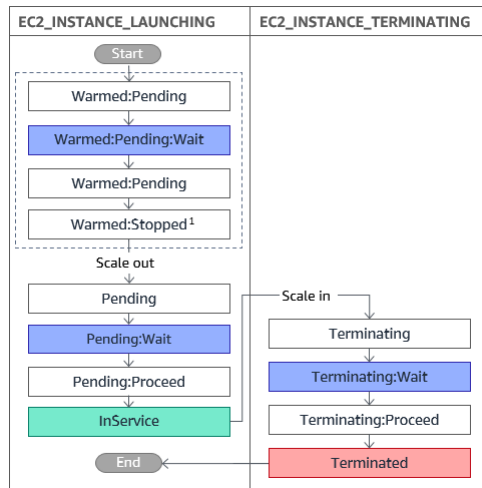
¹ If you opted to put warmed instances into a `Running` state, instances in the warmed pool transition from `Warmed:Pending` to `Warmed:Running` instead.

As shown in the preceding diagram:

- A lifecycle hook runs immediately after an instance is launched into the warm pool, when it transitions to the `Warmed:Pending:Wait` state.
- Another lifecycle hook runs immediately before an instance enters the `InService` state, when it transitions to the `Pending:Wait` state.
- A final lifecycle hook runs immediately before the instance is terminated, when it transitions to the `Terminating:Wait` state.

When you add lifecycle hooks, consider the following:

- When you add a launch lifecycle hook, Amazon EC2 Auto Scaling pauses an instance that transitions to the `Warmed:Pending:Wait` state and the `Pending:Wait` state.
- If the demand on your application depletes the warm pool, Amazon EC2 Auto Scaling can launch instances directly into the Auto Scaling group if the group has not yet reached its maximum capacity. In this case, your launch lifecycle hook pauses the instance only at the `Pending:Wait` state.

These are some of the reasons that you might create lifecycle actions for instances leaving the warm pool:

- You can use this time to prepare EC2 instances for use, for example, if you have services that must start for your application to work correctly.
- You want to pre-populate cache data to ensure that a new server doesn't launch with a completely empty cache.
- You want to register new instances as managed instances with your configuration management service.

Supported notification targets

Amazon EC2 Auto Scaling can send lifecycle event notification messages to the following notification targets.

- Lambda functions (via EventBridge)

- Amazon SNS topics
- Amazon SQS queues

The following information can help you configure these notification targets when creating lifecycle hooks based on the following procedures:

EventBridge rules: With Amazon EventBridge, you can create rules that trigger programmatic actions in response to warm pool events. For example, you can create two EventBridge rules, one for when an instance is entering a warm pool, and one for when an instance is leaving the warm pool.

The EventBridge rules can trigger a Lambda function to perform programmatic actions on instances, depending on your use case and application. The Lambda function is invoked with an incoming event emitted by the Auto Scaling group. Lambda knows which Lambda function to invoke based on the event pattern in the EventBridge rule.

For more information, see .

Amazon SNS notification targets: You can add lifecycle hooks to receive Amazon SNS notifications when a lifecycle action occurs. These notifications are sent to recipients who are subscribed to the Amazon SNS topic that you specify.

Before you add a lifecycle hook that notifies an Amazon SNS topic, you must have already set up the topic and an IAM service role that gives Amazon EC2 Auto Scaling permission to publish to Amazon SNS.

To receive separate notifications about instances leaving and entering the warm pool, you must also have set up Amazon SNS filtering.

Amazon SQS notification targets: You can add lifecycle hooks to an Auto Scaling group and use Amazon SQS to set up a notification target to receive notifications when a lifecycle action occurs. These notifications are sent to the Amazon SQS queue that you specify.

Before you add a lifecycle hook that notifies an Amazon SQS queue, you must have already set up the queue and an IAM service role that gives Amazon EC2 Auto Scaling permission to send notification messages to Amazon SQS.

If you want the queue consumer to process separate notifications about instances leaving but not entering the warm pool (or vice versa), you must also have set up the queue consumer to parse the message and then act on the message if a specific attribute matches the desired value.

Event types and event patterns that you use when you add or update lifecycle hooks

When you add lifecycle hooks to your Auto Scaling group, events are sent to EventBridge in JSON format. You can create an EventBridge rule that uses an event pattern to filter incoming events and then invokes your Lambda function or other target.

There are two types of events that are emitted for lifecycle hooks:

- `EC2 Instance-launch Lifecycle Action`
- `EC2 Instance-terminate Lifecycle Action.`

After you add a warm pool to your Auto Scaling group, the `detail` section for `EC2 Instance-launch Lifecycle Action` events contains new `Origin` and `Destination` fields.

The values of `Origin` and `Destination` can be the following:

`EC2` | `AutoScalingGroup` | `WarmPool`

Contents

Warm pool events

This section lists example events from Amazon EC2 Auto Scaling. Events are emitted on a best-effort basis.

Instances bound for the warm pool

The following example shows an event for a lifecycle hook for the `Warmed:Pending:Wait` state. It represents an instance that is launching into the warm pool.

```
{
  "version": "0",
  "id": "18b2ec17-3e9b-4c15-8024-ff2e8ce8786a",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook-for-warming-instances",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

Instances leaving the warm pool

The following example shows an event for a lifecycle hook for the `Pending:Wait` state. It represents an instance that is leaving the warm pool due to a scale-out event.

```
{
  "version": "0",
  "id": "5985cdde-1f01-9ae2-1498-3c8ea162d141",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
  "region": "us-west-2",
```

```
    "resources": [
      "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
    ],
    "detail": {
      "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
      "AutoScalingGroupName": "my-asg",
      "LifecycleHookName": "my-lifecycle-hook-for-warmed-instances",
      "EC2InstanceId": "i-1234567890abcdef0",
      "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
      "NotificationMetadata": "additional-info",
      "Origin": "WarmPool",
      "Destination": "AutoScalingGroup"
    }
}
```

Instances launching outside of the warm pool

The following example shows an event for a lifecycle hook for the `Pending:Wait` state. It represents an instance that is launching directly into the Auto Scaling group.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-02-01T17:18:06.082Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:autoscaling:us-west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-21098EXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook-for-launching-instances",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

Example event patterns

The preceding section provides example events emitted by Amazon EC2 Auto Scaling.

EventBridge event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

The following fields in the event form the event pattern that is defined in the rule to invoke an action:

`"source": "aws.autoscaling"`

Identifies that the event is from Amazon EC2 Auto Scaling.

`"detail-type": "EC2 Instance-launch Lifecycle Action"`

Identifies the event type.

```
"Origin": "EC2"
```

Identifies where the instance is coming from.

```
"Destination": "WarmPool"
```

Identifies where the instance is going to.

Use the following sample event pattern to capture all events that are associated with instances entering the warm pool.

```
{
  "source": [
      "aws.autoscaling"
  ],
  "detail-type": [
      "EC2 Instance-launch Lifecycle Action"
  ],
  "detail": {
      "Origin": [
          "EC2"
      ],
      "Destination": [
          "WarmPool"
      ]
  }
}
```

Use the following sample event pattern to capture all events that are associated with instances leaving the warm pool due to a scale-out event.

```
{
  "source": [
      "aws.autoscaling"
  ],
  "detail-type": [
      "EC2 Instance-launch Lifecycle Action"
  ],
  "detail": {
      "Origin": [
          "WarmPool"
      ],
      "Destination": [
          "AutoScalingGroup"
      ]
  }
}
```

Use the following sample event pattern to capture all events that are associated with instances launching directly into the Auto Scaling group.

```
{
  "source": [
      "aws.autoscaling"
  ],
  "detail-type": [
      "EC2 Instance-launch Lifecycle Action"
  ],
  "detail": {
      "Origin": [
          "EC2"
```

```
        ],
        "Destination": [
            "AutoScalingGroup"
        ]
    }
}
```

Use the following sample event pattern to capture all events that are associated with `EC2 Instance-launch Lifecycle Action`, regardless of the origin or destination.

```
{
  "source": [
      "aws.autoscaling"
  ],
  "detail-type": [
      "EC2 Instance-launch Lifecycle Action"
  ]
}
```

Creating EventBridge rules for warm pool events

The following procedures explain how to create an EventBridge rule for warm pool events. This sample rule detects events that use the event pattern for instances entering the warm pool, and then sends those events to an AWS Lambda function for processing. A Lambda target is the subject of this procedure, but rules can invoke many types of targets. For information about supported targets, see Amazon EventBridge targets in the *Amazon EventBridge User Guide*.

Before you create the rule, create the AWS Lambda function that you want the rule to use as a target. When you create the rule, you'll need to specify this function as the target for the rule. For information about creating Lambda functions, see Getting started with Lambda in the *AWS Lambda Developer Guide*. For an introductory tutorial-style guide for creating lifecycle hooks, see Tutorial: Configure a lifecycle hook that invokes a Lambda function (p. 168).

Create an EventBridge rule (console)

To create an event rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, under **Events**, choose **Rules**.
3. In the **Rules** section, choose **Create rule**.
4. For **Name**, enter a name for the rule. Optionally enter a description of the rule in the **Description** box.
5. Under **Define pattern**, choose **Event pattern**.
6. For **Event matching pattern**, choose **Custom pattern**.
7. Rules use event patterns to select events and route them to targets. Copy the following pattern into the **Event pattern** box.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
  }
}
```

8. To save the event pattern, choose **Save**.

9. For **Select event bus**, choose **AWS default event bus**.

10. Under **Select targets**, for **Target**, choose **Lambda function**. Then, for **Function**, choose the function that you want to send the events to.

11. When you finish entering settings for the rule, choose **Create**.

12. After you have followed these instructions, continue on to Adding lifecycle hooks (p. 166) as a next step.

Create an EventBridge rule (AWS CLI)

If you haven't already, create the Lambda function that you want the rule to use as a target. When you create the function, note the Amazon Resource Name (ARN) of the function. You'll need to enter this ARN when you specify the target for the rule.

To create an event rule

1. To create a rule that matches events for instances entering the warm pool, use the following put-rule command.

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state ENABLED
```

The following example shows the `pattern.json` to match the event for instances entering the warm pool.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
    }
}
```

If the command runs successfully, EventBridge responds with the ARN of the rule. Note this ARN. You'll need to enter it in step 3.

2. To specify the Lambda function to use as a target for the rule, use the following put-targets command.

```
aws events put-targets --rule my-rule --targets
 Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

In the preceding command, *my-rule* is the name that you specified for the rule in step 1, and the value for the `Arn` parameter is the ARN of the function that you want the rule to use as a target.

3. To add permissions that allow the rule to invoke the target Lambda function, use the following Lambda add-permission command.

```
aws lambda add-permission --function-name my-function --statement-
id AllowInvokeFromExampleEvent \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
 arn:aws:events:region:123456789012:rule/my-rule
```

In the preceding command:

- *my-function* is the name of the Lambda function that you want the rule to use as a target.

Amazon EC2 Auto Scaling User Guide
Viewing health check status and
the reason for health check failures

- *AllowInvokeFromExampleEvent* is a unique identifier that you define to describe the statement in the Lambda function policy.
- `source-arn` is the ARN of the EventBridge rule.

If the command runs successfully, you receive output similar to the following.

```
{
  "Statement": "{\"Sid\":\"AllowInvokeFromExampleEvent\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
          \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

The `Statement` value is a JSON string version of the statement that was added to the Lambda function policy.

4. After you have followed these instructions, continue on to as a next step.

Viewing health check status and the reason for health check failures

Health checks allow Amazon EC2 Auto Scaling to determine when an instance is unhealthy and should be terminated. For warm pool instances kept in a `Stopped` state, it employs the knowledge that Amazon EBS has of a `Stopped` instance's availability to identify unhealthy instances. It does this by calling the `DescribeVolumeStatus` API to determine the status of the EBS volume that's attached to the instance. For warm pool instances kept in a `Running` state, it relies on EC2 status checks to determine instance health. While there is no health check grace period for warm pool instances, Amazon EC2 Auto Scaling doesn't start checking instance health until the lifecycle hook finishes.

When an instance is found to be unhealthy, Amazon EC2 Auto Scaling automatically deletes the unhealthy instance and creates a new one to replace it. Instances are usually terminated within a few minutes after failing their health check. For more information, see .

Custom health checks are also supported. This can be helpful if you have your own health check system that can detect an instance's health and send this information to Amazon EC2 Auto Scaling. For more information, see .

On the Amazon EC2 Auto Scaling console, you can view the status (healthy or unhealthy) of your warm pool instances. You can also view their health status using the AWS CLI or one of the SDKs.

To view the status of your warm pool instances (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Instance management** tab, under **Warm pool instances**, the **Lifecycle** column contains the state of your instances.

Amazon EC2 Auto Scaling User Guide
Viewing health check status and
the reason for health check failures

The **Health status** column shows the assessment that Amazon EC2 Auto Scaling has made of instance health.

> **Note**
> New instances start healthy. Until the lifecycle hook is finished, an instance's health is not checked.

To view the reason for health check failures (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Activity** tab, under **Activity history**, the **Status** column shows whether your Auto Scaling group has successfully launched or terminated instances.

   If it terminated any unhealthy instances, the **Cause** column shows the date and time of the termination and the reason for the health check failure. For example, "At 2021-04-01T21:48:35Z an instance was taken out of service in response to EBS volume health check failure".

To view the status of your warm pool instances (AWS CLI)

View the warm pool for an Auto Scaling group by using the following describe-warm-pool command.

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

Example output.

```
{
    "WarmPoolConfiguration": {
        "MinSize": 0,
        "PoolState": "Stopped"
    },
    "Instances": [
        {
            "InstanceId": "i-0b5e5e7521cfaa46c",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        },
        {
            "InstanceId": "i-0e21af9dcfb7aa6bf",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        }
```

```
    ]
}
```

To view the reason for health check failures (AWS CLI)

Use the following describe-scaling-activities command.

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

The following is an example response, where `Description` indicates that your Auto Scaling group has terminated an instance and `Cause` indicates the reason for the health check failure.

Scaling activities are ordered by start time. Activities still in progress are described first.

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in response to
 EBS volume health check failure.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

Deleting a warm pool

When you no longer need the warm pool, use the following procedure to delete it.

To delete your warm pool (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to an existing group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. Choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Delete**.

To delete your warm pool (AWS CLI)

Use the following delete-warm-pool command to delete the warm pool.

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

If there are instances in the warm pool, or if scaling activities are in progress, use the delete-warm-pool command with the `--force-delete` option. This option will also terminate the Amazon EC2 instances and any outstanding lifecycle actions.

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

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 the following about Auto Scaling groups with a mixed instances policy (p. 53):

- Amazon EC2 Auto Scaling first identifies which of the two types (Spot or On-Demand) should be terminated. It then applies the termination policy in each Availability Zone individually, and identifies which instance (within the identified purchase option) in which Availability Zone to terminate that will result in the Availability Zones being most balanced. The same principles apply to Auto Scaling groups that use a mixed instances configuration with weights defined for the instance types.

Contents

Default termination policy

The default termination policy is designed to help ensure that your instances span Availability Zones evenly for high availability (p. 6). The default policy is kept generic and flexible to cover a range of scenarios.

Before Amazon EC2 Auto Scaling selects an instance to terminate, it first determines which Availability Zones have the most instances, and at least one instance that is not protected from scale in.

Within the selected Availability Zone, the default termination policy behavior is as follows:

1. 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 a mixed instances policy, which uses allocation strategies (p. 54).

   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.
2. 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.
3. 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.

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 desired 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 termination policies:

- `Default`. Terminate instances according to the default termination policy. This policy is useful when you want your Spot allocation strategy evaluated before any other policy, so that every time your Spot instances are terminated or replaced, you continue to make use of Spot Instances in the optimal pools. It is also useful, for example, when you want to move off launch configurations and start using launch templates.
- `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.

- `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.

- `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.

- `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.

- `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.

  Note
  Amazon EC2 Auto Scaling always balances instances across Availability Zones first, regardless of which termination policy is used. As a result, you might encounter situations in which some newer instances are terminated before older instances when there is a more recently added Availability Zone, or when one Availability Zone has more instances than the other Availability Zones that are used by the group.

To customize a termination policy (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Advanced configurations**, **Edit**.
4. For **Termination policies**, 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.
5. Choose **Update**.

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. 205). To prevent Amazon EC2 Auto Scaling from terminating unhealthy instances, suspend the `ReplaceUnhealthy` process. For more information, see Suspending and resuming scaling processes (p. 199).
- 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 (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 (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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select check box next to the Auto Scaling group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Advanced configurations**, **Edit**.
4. For **Instance scale-in protection**, select **Enable instance scale-in protection**.
5. Choose **Update**.

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Instance management** tab, in **Instances**, select an instance.
4. To enable instance scale-in protection, choose **Actions**, **Set scale-in protection**. When prompted, choose **Set scale-in protection**.
5. 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
```

Common termination policy scenarios for Amazon EC2 Auto Scaling

The following are common termination policy scenarios for Amazon EC2 Auto Scaling instance termination. By default, Amazon EC2 Auto Scaling uses the default termination policy, but you can optionally specify a termination policy of your own.

Contents

Scale-in events

The most common scenario for using a termination policy is when Amazon EC2 Auto Scaling launches Amazon EC2 instances using scaling policies, and then terminates those instances when they are no longer needed as part of a scale-in event. A scale-in event can also occur because of a scheduled action or where there is a new value for desired capacity that is lower than the current capacity of the group.

Consider an Auto Scaling group that has one instance type, two Availability Zones, a desired capacity of two instances, and a scaling policy that adds and removes instances as resource utilization increases and decreases. The two instances in this group are distributed as follows.
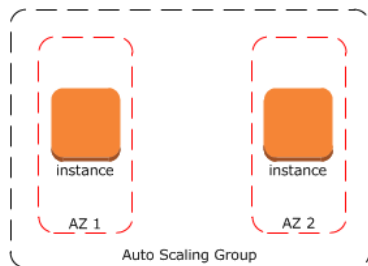
When the Auto Scaling group scales out, Amazon EC2 Auto Scaling launches a new instance. The Auto Scaling group now has three instances, distributed as follows.
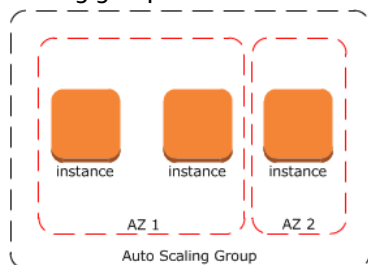
When the Auto Scaling group scales in, Amazon EC2 Auto Scaling terminates one of the instances.

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 that was launched from the oldest launch configuration. If the instances were launched from the same launch configuration, Amazon EC2 Auto Scaling selects the instance that is closest to the next billing hour and terminates it.

Note that the default termination policy works for Auto Scaling groups created with a launch template or with a launch configuration. You can move your Auto Scaling groups from using launch configurations to launch templates at any time and continue to use the default termination policy. The default termination policy will continue to terminate instances launched from the oldest launch configuration until there are no more remaining instances created from a launch configuration. After that, it terminates instances launched from the oldest launch template.

Rebalancing across Availability Zones

occur to proactively balance your instances across Availability Zones evenly for high availability. For example, rebalancing can be necessary when there's an availability outage, when there are changes to the Availability Zones, and when you remove instances. When terminating instances due to rebalancing activities, the termination policy determines which instances are terminated.

Availability outage

Availability outages are rare. However, if one Availability Zone becomes unavailable, and then recovers, your Auto Scaling group can become unbalanced between Availability Zones. Amazon EC2 Auto Scaling then tries to gradually rebalance the group, and rebalancing might terminate instances in other zones.

Take the example where you have an Auto Scaling group that has one instance type, two Availability Zones, and a desired capacity of two instances. In a situation where one Availability Zone fails, Amazon EC2 Auto Scaling automatically launches a new instance in the healthy Availability Zone to replace the one in the unhealthy Availability Zone.

When the unhealthy Availability Zone returns to a healthy state, Amazon EC2 Auto Scaling automatically launches a new instance in this zone, which in turn terminates an instance in the unaffected zone.

Changes to Availability Zones

An existing Auto Scaling group can be updated to add more subnets, either for existing Availability Zones or for new Availability Zones that have been added since the creation of the Auto Scaling group. If you expand your Auto Scaling group to include additional Availability Zones, or you change which Availability Zones are used, Amazon EC2 Auto Scaling will launch instances in the new Availability Zones and terminate instances in the other zones to help ensure that your instances span Availability Zones evenly.

Removing instances

If you detach instances from your Auto Scaling group, or you explicitly terminate instances and decrement the desired capacity, thereby preventing replacement instances from launching, the group can become unbalanced. If this occurs, Amazon EC2 Auto Scaling compensates by rebalancing the Availability Zones.

Instance refreshes

During an instance refresh, which you start in order to update the instances in your Auto Scaling group, Amazon EC2 Auto Scaling terminates instances in the group and then launches replacements for the terminated instances.

For example, let's say that you changed the instance types that are associated with your Auto Scaling group. After making the change, you start an instance refresh to force replacement instances to launch that use your new instance types. If you run an instance refresh on a group of 10 instances and set the

minimum healthy percentage to 90%, Amazon EC2 Auto Scaling replaces one instance before continuing on to replace the next instance. The termination policy controls which instance is replaced first.

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. 188).
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:
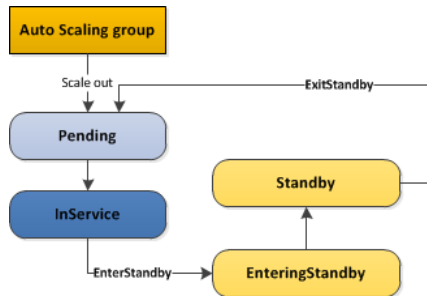
For more information about the complete lifecycle of instances in an Auto Scaling group, see Amazon EC2 Auto Scaling instance lifecycle (p. 7).

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Instance management** tab, in **Instances**, select an instance.
4. Choose **Actions**, **Set to Standby**.
5. 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**.
6. You can update or troubleshoot your instance as needed. When you have finished, continue with the next step to return the instance to service.
7. 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. 195). 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. 191).

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 (p. 199)
- Choosing to suspend (p. 200)
- Suspend and resume scaling processes (console) (p. 202)
- Suspend and resume scaling processes (AWS CLI) (p. 202)

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. For more information, see Elastic Load Balancing and Amazon EC2 Auto Scaling (p. 86).
- `AlarmNotification`—Accepts notifications from CloudWatch alarms that are associated with the group's scaling policies. For more information, see Dynamic scaling for Amazon EC2 Auto Scaling (p. 123).
- `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. 6).
- `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. 205).

- `InstanceRefresh`—Terminates and replaces instances using the instance refresh feature. For more information, see Replacing Auto Scaling instances based on an instance refresh (p. 100).

- `ReplaceUnhealthy`—Terminates instances that are marked as unhealthy and then creates new instances to replace them. For more information, see Health checks for Auto Scaling instances (p. 205).

- `ScheduledActions`—Performs the scheduled scaling actions that you create or that are created by the predictive scaling feature of AWS Auto Scaling. For more information, see Scheduled scaling for Amazon EC2 Auto Scaling (p. 153).

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.

  - `InstanceRefresh` is paused until you resume `Terminate`.

`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.

- `InstanceRefresh` is paused until you resume `Launch`.

`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.
- The instance refresh feature cannot replace any instances.

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.

- **Maximum instance lifetime**—When `Launch` or `Terminate` are suspended, the maximum instance lifetime feature cannot replace any instances.
- **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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to the Auto Scaling group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Advanced configurations**, **Edit**.
4. For **Suspended processes**, choose the process to suspend.

    To resume a suspended process, remove it from **Suspended processes**.
5. Choose **Update**.

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. AWS provides the following 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. If an instance does not pass its health check, it is marked unhealthy and will be terminated while Amazon EC2 Auto Scaling launches a new instance in replacement. For more information, see Health checks for Auto Scaling instances (p. 205).

Capacity Rebalancing

You can enable Capacity Rebalancing on new and existing Auto Scaling groups when using Spot Instances. When you turn on Capacity Rebalancing, Amazon EC2 Auto Scaling attempts to launch a Spot Instance whenever Amazon EC2 notifies that a Spot Instance is at an elevated risk of interruption. After launching a new instance, it then terminates an old instance. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

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. 214).

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 Monitoring CloudWatch metrics for your Auto Scaling groups and instances (p. 215).

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. 222).

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.

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. 224).

EventBridge

Amazon EventBridge, formerly called CloudWatch Events, delivers a near real-time stream of system events that describe changes in AWS resources. EventBridge enables automated event-driven computing, as you can write rules that watch for certain events and trigger automated actions in other AWS services when these events happen. For example, you can use EventBridge to set up a target to invoke a Lambda function when your Auto Scaling group scales or when a lifecycle action occurs. You can also receive a two-minute warning when Spot Instances are about to be reclaimed by Amazon EC2. For information about capturing Amazon EC2 Auto Scaling emitted events in EventBridge, see Using Amazon EC2 Auto Scaling with EventBridge (p. 227).

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 (ELB), 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. The default health checks for an Auto Scaling group are EC2 status checks only.
- Health checks provided by Elastic Load Balancing (ELB). These health checks are disabled by default but can be enabled.
- Your custom health checks.

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 all 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 Status Checks

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`
- `shutting-down`
- `terminated`

The EC2 status checks do not require any special configuration and are always enabled. 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*.

Elastic Load Balancing (`ELB`) Health Checks

Instances for groups that do not use `ELB` health checks are considered healthy if they are in the `running` state. Instances for groups that use `ELB` health checks are considered healthy if they are in the `running` state and they are reported as healthy by the load balancer.

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 `unhealthy`. 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. For more information, see Adding ELB health checks (p. 91).

Custom Health Checks

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. For more information, see Using custom health checks (p. 207).

Health check grace period

When an instance launches, Amazon EC2 Auto Scaling uses the value of the `HealthCheckGracePeriod` for the Auto Scaling group to determine how long to wait before checking the health status of the instance. Amazon EC2 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.

By default, the health check grace period is `300` seconds when you create an Auto Scaling group from the AWS Management Console. Its default value is `0` seconds when you create an Auto Scaling group using the AWS CLI or an SDK.

To provide ample warm-up time for your instances, ensure that the health check grace period covers the expected startup time for your application, from when an instance comes into service to when it can

receive traffic. 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.

Replacing unhealthy instances

After an instance has been marked unhealthy because of a health check, it is almost immediately scheduled for replacement. It never automatically recovers its health. You can intervene manually by calling the **set-instance-health** command or the `SetInstanceHealth` operation to set the instance's health status back to healthy. If the instance is already terminating, you get an error.

> **Note**
> 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 **set-instance-health** command or the `SetInstanceHealth` operation is probably useful only for a suspended group. For more information, see Suspending and resuming scaling processes (p. 199).

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 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"
            },
            ...
        ]
    }
  ]
}
```

See also

For more information about health checks, see Troubleshooting Amazon EC2 Auto Scaling: Health checks (p. 279). If your health checks fail, check this topic for troubleshooting steps. This topic will help you figure out what has gone wrong in your Auto Scaling group and give you suggestions on how to fix it.

Amazon EC2 Auto Scaling Capacity Rebalancing

You can configure Amazon EC2 Auto Scaling to monitor and automatically respond to changes that affect the availability of your Spot Instances. Capacity Rebalancing helps you maintain workload availability by proactively augmenting your fleet with a new Spot Instance before a running instance is interrupted by EC2.

How it works

Amazon EC2 Auto Scaling is aware of EC2 instance rebalance recommendation notifications. The Amazon EC2 Spot service emits these notifications when Spot Instances are at elevated risk of interruption. When Capacity Rebalancing is enabled for an Auto Scaling group, Amazon EC2 Auto Scaling attempts to proactively replace Spot Instances in the group that have received a rebalance recommendation, providing the opportunity to rebalance your workload to new Spot Instances that are not at elevated risk of interruption. This means that your workload can continue to process the work while Amazon EC2 Auto Scaling launches a new Spot Instance before an existing instance is interrupted. You can also optionally use a lifecycle hook to perform a custom action on instances before termination.

For more information about EC2 instance rebalance recommendations, see EC2 instance rebalance recommendations in the *Amazon EC2 User Guide for Linux Instances*.

For more details and a walkthrough of the Capacity Rebalancing feature, see the blog post Proactively manage Spot Instance lifecycle using the new Capacity Rebalancing feature for EC2 Auto Scaling on the AWS Compute Blog.

> **Note**
> When Capacity Rebalancing is disabled, Amazon EC2 Auto Scaling replaces Spot Instances after the Amazon EC2 Spot service interrupts the instances and their health check fails. Amazon EC2 always gives both an EC2 instance rebalance recommendation and a Spot two-minute instance interruption notice before an instance is interrupted.

Contents

Enabling Capacity Rebalancing

You can enable or disable Capacity Rebalancing at any time.

Considerations

The following considerations apply for this configuration:

- We recommend that you configure your Auto Scaling group to use multiple instance types. This provides the flexibility to launch instances in various Spot Instance pools within each Availability Zone, as documented in Auto Scaling groups with multiple instance types and purchase options (p. 53).
- We also recommend that you use the `capacity-optimized` allocation strategy. Used in combination with the Capacity Rebalancing feature, the `capacity-optimized` allocation strategy ensures that your instances are maintained in the most optimal Spot pools.
- Whenever possible, you should create your Auto Scaling group in all Availability Zones within the Region. This is to enable Amazon EC2 Auto Scaling to look at the free capacity in each Availability Zone. If a launch fails in one Availability Zone, Amazon EC2 Auto Scaling keeps trying to launch Spot Instances across the specified Availability Zones until it succeeds.
- Using Capacity Rebalancing, Amazon EC2 Auto Scaling behaves in the following way: When launching a new instance, Amazon EC2 Auto Scaling waits until the new instance passes its health check before it proceeds with terminating the old instance. When replacing more than one instance, the termination of each old instance starts after the new instance has launched and passed its health check. Because Amazon EC2 Auto Scaling attempts to launch new instances before terminating old ones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities. To avoid this problem, Amazon EC2 Auto Scaling can temporarily exceed the specified maximum capacity of a group during a rebalancing activity.
  - If the new instances fail to launch or they launch but the health check fails, Amazon EC2 Auto Scaling keeps trying to relaunch them. While it is trying to launch new instances, your old ones will eventually be interrupted and forcibly terminated.
  - If a scaling activity is in progress and your Auto Scaling group is below its new desired capacity, Amazon EC2 Auto Scaling scales out first before terminating the old instances.
- You can configure a lifecycle hook when enabling Capacity Rebalancing to gracefully shut down your application inside the instances that receive the rebalance notification, before Amazon EC2 Auto Scaling terminates the instances.
- When using Spot Instances, your application should be tolerant of some interruption to prevent disruptions to your application. In cases where an instance receives a final two-minute interruption notice, Amazon EC2 Auto Scaling calls the terminating lifecycle hook and attempts to launch a replacement immediately.

Enabling Capacity Rebalancing (console)

You can enable or disable Capacity Rebalancing when you create or update an Auto Scaling group.

To enable Capacity Rebalancing for a new Auto Scaling group

Follow the instructions in Auto Scaling groups with multiple instance types and purchase options (p. 53) to create a new Auto Scaling group. When you create an Auto Scaling group, in step 2 of the wizard, you configure the instance purchase options and network settings. To specify additional settings for your Auto Scaling group, including instance distribution settings, Spot allocation settings, Capacity

Rebalancing, and the instance types that Amazon EC2 Auto Scaling can launch, select **Combine purchase options and instance types**.

Under the **Instances distribution** section, you can select whether or not to enable Capacity Rebalancing by selecting or clearing the **Capacity rebalance** check box. This setting is enabled by default on Auto Scaling groups created using the console when the Spot allocation strategy is set to **Capacity optimized**.

To enable Capacity Rebalancing for an existing Auto Scaling group

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Purchase options and instance types**, **Edit**.
4. Under the **Instances distribution** section, do the following:

   - To enable Capacity Rebalancing, select the **Capacity rebalance** check box.
   - To disable Capacity Rebalancing, clear the **Capacity rebalance** check box.
5. Choose **Update**.

Enabling Capacity Rebalancing (AWS CLI)

The following examples show how to use the AWS CLI to enable and disable Capacity Rebalancing.

Use the **create-auto-scaling-group** or **update-auto-scaling-group** command with the following parameter:

- `--capacity-rebalance` / `--no-capacity-rebalance` — Boolean value that indicates whether Capacity Rebalancing is enabled.

Before you call the **create-auto-scaling-group** command, you need the name of a launch template that is configured for use with an Auto Scaling group. For more information, see Creating a launch template for an Auto Scaling group (p. 26).

Note

The following procedures show how to use a configuration file formatted in JSON or YAML. If you use AWS CLI version 1, you must specify a JSON-formatted configuration file. If you use AWS CLI version 2, you can specify a configuration file formatted in either YAML or JSON.

JSON

To create and configure a new Auto Scaling group

- Use the following **create-auto-scaling-group** command to create a new Auto Scaling group and enable Capacity Rebalancing, referencing a JSON file as the sole parameter for your Auto Scaling group.

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

If you don't already have a CLI configuration file that specifies a mixed instances policy (p. 53), create one.

Add the following line to the top-level JSON object in the configuration file.

```
{
    "CapacityRebalance": true
}
```

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

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacity": 12,
    "MinSize": 12,
    "MaxSize": 15,
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 0,
            "OnDemandPercentageAboveBaseCapacity": 25,
            "SpotAllocationStrategy": "capacity-optimized"
        },
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
```

```
            },
            {
                    "InstanceType": "c3.large"
            },
            {
                    "InstanceType": "m3.large"
            }
        ]
    }
},
"TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
"VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

To create and configure a new Auto Scaling group

- Use the following **create-auto-scaling-group** command to create a new Auto Scaling group and enable Capacity Rebalancing, referencing a YAML file as the sole parameter for your Auto Scaling group.

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

Add the following line to your configuration file formatted in YAML.

```
CapacityRebalance: true
```

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

```
---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
TargetGroupARNs:
- arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-
group/943f017f100becff
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

To enable Capacity Rebalancing for an existing Auto Scaling group

- Use the following **update-auto-scaling-group** command to enable Capacity Rebalancing.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

To verify that Capacity Rebalancing is enabled for an Auto Scaling group

- Use the following **describe-auto-scaling-groups** command to verify that Capacity Rebalancing is enabled and to view the details.

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

The following is an example response.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "CapacityRebalance": true
        }
    ]
}
```

To disable Capacity Rebalancing

Use the **update-auto-scaling-group** command with the `--no-capacity-rebalance` option to disable Capacity Rebalancing.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --no-capacity-rebalance
```

Adding a termination lifecycle hook

Consider configuring a termination lifecycle hook when enabling Capacity Rebalancing so that you can run code and perform custom actions on an instance before it is terminated.

Reasons you might use a termination lifecycle hook include:

- To upload system or application logs to Amazon Simple Storage Service (Amazon S3)
- For graceful shutdown of Amazon SQS workers
- To complete deregistration from the Domain Name System (DNS)

If you don't have a termination lifecycle hook, use the following procedure to create one.

To add a termination lifecycle hook

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

A split pane opens in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.

3. On the **Instance management** tab, in **Lifecycle hooks**, choose **Create lifecycle hook**.

4. 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 terminate**.

   c. For **Heartbeat timeout**, specify the amount of time, in seconds, that you will have to complete the lifecycle action, or until the timeout period ends. We recommend a value from 30 to 120 seconds, depending on how long you need to shut down your application.

   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. Both ABANDON and CONTINUE allow the instance to terminate.

      • If you choose CONTINUE, the Auto Scaling group can proceed with any remaining actions, such as other lifecycle hooks, before termination.

      • If you choose ABANDON, the Auto Scaling group terminates the instances immediately.

   e. (Optional) For **Notification metadata**, specify additional information that you want to include anytime that Amazon EC2 Auto Scaling sends a message to an AWS Lambda function or another target that you configure in step 7.

5. Choose **Create**.

6. (Optional) To add an action to your lifecycle hook, follow these steps (p. 162) to configure an AWS Lambda function or another target. Otherwise, for an EC2 instance to run an action automatically, you must configure it to run scripts. We recommend that you script your shutdown sequence to be completed in under one to two minutes to ensure that there is enough time to complete tasks before instance termination.

For more information, see Amazon EC2 Auto Scaling lifecycle hooks (p. 158).

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 set up 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
```

Monitoring CloudWatch metrics for your Auto Scaling groups and instances

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. You can enable and disable these metrics using the AWS Management Console, AWS CLI, or an SDK.

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. For more information, see Configuring monitoring for Auto Scaling instances (p. 221).

Contents

Enable Auto Scaling group metrics (console)

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

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the page, showing information about the group that's selected.
3. On the **Monitoring** tab, select the **Auto Scaling group metrics collection**, **Enable** check box located at the top of the page under **Auto Scaling**.

To disable group metrics

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select your Auto Scaling group.
3. On the **Monitoring** tab, clear the **Auto Scaling group metrics collection**, **Enable** check box.

Enable Auto Scaling group metrics (AWS CLI)

To enable group metrics

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

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
```

Available metrics and dimensions

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. **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 Auto Scaling groups that use the instance weighting (p. 69) feature. If instance weighting is not applied, then the following metrics are populated, but are equal to the metrics that are defined in the preceding table.

Metric	Description
GroupInServiceCapacity	The number of capacity units that are running as part of the Auto Scaling group.

Amazon EC2 Auto Scaling User Guide
Viewing graphed metrics for your
Auto Scaling groups and instances

Metric	Description
	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.
`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.

Viewing graphed metrics for your Auto Scaling groups and instances

After you create an Auto Scaling group, you can open the group and view a series of monitoring graphs on the **Monitoring** tab. Each graph is based on one of the available CloudWatch metrics for your Auto Scaling groups and instances. The monitoring graphs show data points for Auto Scaling group metrics if the metrics are enabled.

The following graphed 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`

The following graphed metrics are available for groups where instances have weights that define how many units each instance contributes to the desired capacity of the group:

- **In Service Capacity Units** — `GroupInServiceCapacity`
- **Pending Capacity Units** — `GroupPendingCapacity`
- **Standby Capacity Units** — `GroupStandbyCapacity`
- **Terminating Capacity Units** — `GroupTerminatingCapacity`
- **Total Capacity Units** — `GroupTotalCapacity`

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`

For more information about the Amazon EC2 metrics and the data they provide to the graphs, see List the available CloudWatch metrics for your instances in the *Amazon EC2 User Guide for Linux Instances*.

Working with Amazon CloudWatch

Contents
- View CloudWatch metrics (p. 219)
- Create Amazon CloudWatch alarms (p. 219)

View CloudWatch metrics

You can view your Auto Scaling group metrics using the CloudWatch console and the command line tools.

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. In Amazon CloudWatch, you can create an alarm that sends a notification when the value of a certain metric is beyond what you consider an acceptable threshold.

Start by identifying the metric to monitor. For example, you can configure an alarm to watch over the average CPU utilization of the EC2 instances in your Auto Scaling group. The action can be a notification that is sent to you when the average CPU utilization of the group's instances 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 for your Auto Scaling group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. If necessary, change the region. From the navigation bar, select the region where your Auto Scaling group resides.

3. On the navigation pane, choose **Alarms** and then choose **Create alarm**.

4. Choose **Select metric**.

5. On the **All metrics** tab, select a metric as follows:

   - To display only the metrics reported for your Auto Scaling groups, choose **EC2**, and then choose **By Auto Scaling Group**. To view the metrics for a single Auto Scaling group, type its name in the search field.

   - Select the row that contains the metric for the Auto Scaling group that you want to create an alarm on.

   - Choose **Select metric**. The **Specify metric and conditions** page appears, showing a graph and other information about the metric.

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. Under **Conditions**, do the following:

   - For **Threshold type**, choose **Static**.

   - For **Whenever `metric` is**, specify whether you want the value of the metric to be greater than, greater than or equal to, less than, or less than or equal to the threshold to trigger the alarm. Then, under **than**, enter the threshold value that you want to trigger the alarm.

8. Under **Additional configuration**, do the following:

   - For **Datapoints to alarm**, enter the number of data points (evaluation periods) during which the metric value must meet the threshold conditions to trigger the alarm. For example, two consecutive periods of 5 minutes would take 10 minutes to trigger the alarm.

   - For **Missing data treatment**, choose what you want the alarm to do if some data is missing. For more information, see Configuring how CloudWatch alarms treat missing data in the *Amazon CloudWatch User Guide*.

9. Choose **Next**.

10. Under **Notification**, you can choose or create the Amazon SNS topic you want to use to receive notifications. Otherwise, you can remove the notification now and add one later when you are ready.

11. Choose **Next**.

12. Enter a name and, optionally, a description for the alarm, and then choose **Next**.

13. Choose **Create Alarm**.

Configuring monitoring for Auto Scaling instances

Amazon EC2 can enable detailed monitoring when it is launching EC2 instances in your Auto Scaling group. You configure monitoring for Auto Scaling instances using a launch template or launch configuration.

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 and Monitoring your instances using CloudWatch in the *Amazon EC2 User Guide for Linux Instances*.

Enable detailed monitoring (console)

By default, basic monitoring is enabled when you use the AWS Management Console to create a launch template or launch configuration.

To enable detailed monitoring in a launch template

When you create the launch template using the AWS Management Console, in the **Advanced details** section, for **Detailed CloudWatch monitoring**, choose **Enable**. Otherwise, basic monitoring is enabled. For more information, see Configuring advanced settings for your launch template (p. 30).

To enable detailed monitoring in a launch configuration

When you create the launch configuration using the AWS Management Console, in the **Additional configuration** section, select **Enable EC2 instance detailed monitoring within CloudWatch**. Otherwise, basic monitoring is enabled. For more information, see Creating a launch configuration (p. 39).

Enable detailed monitoring (AWS CLI)

By default, basic monitoring is enabled when you create a launch template using the AWS CLI. Detailed monitoring is enabled by default when you create a launch configuration using the AWS CLI.

To enable detailed monitoring in a launch template

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.

To enable detailed monitoring in a launch configuration

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
```

Switching between basic and detailed monitoring

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 template or 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**.

> **Note**
> With the maximum instance lifetime and instance refresh features, you can also replace all instances in the Auto Scaling group to launch new instances that use the new settings. For more

information, see Replacing Auto Scaling instances based on maximum instance lifetime (p. 97) and Replacing Auto Scaling instances based on an instance refresh (p. 100).

When you switch between basic and detailed monitoring:

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.

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"
}
```

Getting Amazon SNS notifications when your Auto Scaling group scales

You can be notified when Amazon EC2 Auto Scaling is launching or terminating the EC2 instances in your Auto Scaling group. You manage notifications using Amazon Simple Notification Service (Amazon SNS).

Amazon SNS coordinates and manages the delivery or sending of notifications to subscribing clients or endpoints. Amazon SNS offers a variety of notification options, including the ability to 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, which enables you to handle these notifications programmatically. For more information, see 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.

Notifications are useful for designing event-driven applications. If you use notifications to check that a resource enters a desired state, you can eliminate polling, and you won't encounter the `RequestLimitExceeded` error that sometimes results from polling.

AWS provides various tools that you can use to send notifications. Alternatively, you can use EventBridge and Amazon SNS to send notifications when your Auto Scaling groups launch or terminate instances. In EventBridge, the rule describes which events you're notified about. In Amazon SNS, the topic describes what kind of notification you receive. Using this option, you can decide if certain events should trigger a Lambda function instead. For more information, see Using Amazon EC2 Auto Scaling with EventBridge (p. 227).

Contents
- SNS notifications (p. 224)
- Configuring Amazon SNS notifications for Amazon EC2 Auto Scaling (p. 225)
    - Create an Amazon SNS topic (p. 225)
    - Subscribe to the Amazon SNS topic (p. 226)
    - Confirm your Amazon SNS subscription (p. 226)
    - Configure your Auto Scaling group to send notifications (p. 226)
    - Test the notification (p. 226)
    - Delete the notification configuration (p. 227)

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"}
```

Configuring Amazon SNS notifications for Amazon EC2 Auto Scaling

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.

When you create a topic name, the name must meet the following requirements:

- Between 1 and 256 characters long
- Contain uppercase and lowercase ASCII letters, numbers, underscores, or hyphens

For more information, see Creating an Amazon SNS 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 Subscribing to an Amazon SNS 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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the page, showing information about the group that's selected.
3. On the **Activity** tab, choose **Activity notifications**, **Create notification**.
4. On the **Create notifications** pane, do the following:

   a. For **SNS Topic**, select your SNS topic.
   b. For **Event types**, select the events to send the notifications.
   c. Choose **Create**.

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

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select the check box next to your Auto Scaling group.

    A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
3. On the **Details** tab, choose **Group details**, **Edit**.
4. 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.
5. Choose **Update**.
6. 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.

Delete the notification configuration

You can delete your Amazon EC2 Auto Scaling notification configuration if it is no longer being used.

To delete Amazon EC2 Auto Scaling notification configuration (console)

1. Open the Amazon EC2 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. Select your Auto Scaling group.
3. On the **Activity** tab, select the check box next to the notification you want to delete and then choose **Actions**, **Delete**.

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 Deleting an Amazon SNS subscription and topic in the *Amazon Simple Notification Service Developer Guide*.

Using Amazon EC2 Auto Scaling with EventBridge

Amazon EventBridge, formerly called CloudWatch Events, helps you set up event-driven rules that monitor resources and initiate target actions that use other AWS services.

Events from Amazon EC2 Auto Scaling are delivered to EventBridge in near real time. You can establish EventBridge rules that trigger programmatic actions and notifications in response to a variety of these events. For example, while instances are in the process of launching or terminating, you can trigger an AWS Lambda function to perform a preconfigured task. Or, you can trigger notifications to an Amazon SNS topic to monitor the progress of an instance refresh and to perform validations at specific checkpoints.

In addition to invoking Lambda functions and notifying Amazon SNS topics, EventBridge supports other types of targets and actions, such as relaying events to Amazon Kinesis streams, activating AWS Step

Functions state machines, and invoking the AWS Systems Manager run command. For information about supported targets, see Amazon EventBridge targets in the *Amazon EventBridge User Guide*.

For more information about EventBridge, see Getting started with Amazon EventBridge in the *Amazon EventBridge User Guide*. Note that you can also create rules that trigger on Amazon EC2 Auto Scaling API calls. For more information, see Creating an EventBridge rule that triggers on an AWS API call using AWS CloudTrail in the *Amazon EventBridge User Guide*.

Contents

Auto Scaling events

The following are example events from Amazon EC2 Auto Scaling. Events are emitted on a best effort basis.

For examples of the events delivered from Amazon EC2 Auto Scaling to EventBridge when you use a warm pool, see Warm pool events (p. 180).

For an example of the event for a Spot Instance interruption, see Spot Instance interruption notices in the *Amazon EC2 User Guide for Linux Instances*.

Events

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"
  }
}
```

EC2 Auto Scaling Instance Refresh Checkpoint Reached

During an instance refresh, Amazon EC2 Auto Scaling emits events when the number of instances that have been replaced reaches the percentage threshold defined for the checkpoint.

Event Data

The following is example data for this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
      "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
      "AutoScalingGroupName": "my-auto-scaling-group",
      "CheckpointPercentage": "50",
      "CheckpointDelay": "300"
  }
}
```

Using AWS Lambda to handle events

AWS Lambda is a compute service that you can use to run code without provisioning or managing servers. You package your code and upload it to AWS Lambda as a *Lambda function*. AWS Lambda then runs the function when the function is invoked. A function can be invoked manually by you, automatically in response to events, or in response to requests from applications or services.

To help you get started with Lambda, follow the procedure below. This section shows you how to create a Lambda function and an EventBridge rule causing all instance launch and terminate events to be logged in Amazon CloudWatch Logs.

For a step-by-step tutorial that shows you how to create a Lambda function to perform lifecycle actions, see Tutorial: Configure a lifecycle hook that invokes a Lambda function (p. 168). This tutorial covers how to get started with lifecycle hooks. With lifecycle hooks, you can use Lambda to perform tasks on instances before they are put into service or before they are terminated.

Create a Lambda function

Use the following procedure to create a Lambda function using the **hello-world** blueprint to serve as the target for events.

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**. For **Existing role**, select your basic execution role. Otherwise, create a basic execution role.

    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

Create a rule that matches selected events and routes them to your Lambda function to take action.

To create a rule that routes events to your Lambda function

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.

2. In the navigation pane, under **Events**, choose **Rules**.

3. Choose **Create rule**.

4. Enter a name and description for the rule.

5. For **Define pattern**, do the following:

    a. Choose **Event Pattern**.

    b. For **Event matching pattern**, choose **Pre-defined by service**.

        Tip
        You can also create a rule that uses a custom event pattern to detect and act upon only a subset of Amazon EC2 Auto Scaling events. This subset can be based on specific fields that Amazon EC2 Auto Scaling includes in its events. For more information, see Event patterns in the *Amazon EventBridge User Guide*. For an example event pattern, see Step 3: Create an EventBridge rule (p. 170) in the tutorial for lifecycle hooks.

    c. For **Service provider**, choose **AWS**.

    d. For **Service Name**, choose **Auto Scaling**.

    e. For **Event type**, choose **Instance Launch and Terminate**.

    f. To capture all successful and unsuccessful instance launch and terminate events, choose **Any instance event**.

    g. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific Auto Scaling group, choose **Specific group name(s)** and select one or more Auto Scaling groups.

6. For **Select event bus**, choose **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.

7. For **Target**, choose **Lambda function**.

8. For **Function**, select the Lambda function that you created.

9. Choose **Create**.

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 Auto Scaling console at https://console.aws.amazon.com/ec2autoscaling/.
2. On the **Details** tab, choose **Edit** from the right side of the page.
3. Change the value of **Desired capacity**, and then choose **Update**.
4. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
5. On the navigation pane, choose **Logs**.
6. Select the log group for your Lambda function (for example, **/aws/lambda/**_my-function_).
7. 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",
```

Creating EventBridge rules for instance refresh events

This section shows you how to create a rule that notifies you whenever a checkpoint is reached during an instance refresh. The procedure for setting up email notifications through Amazon SNS is included. 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 that your Auto Scaling group uses to send the notifications. You create a topic by specifying a name for your topic.

When you create a topic name, the name must meet the following requirements:

- Between 1 and 256 characters long
- Contain uppercase and lowercase ASCII letters, numbers, underscores, or hyphens

For more information, see Creating an Amazon SNS 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 Subscribing to an Amazon SNS 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.

Route events to your Amazon SNS topic

Create a rule that matches selected events and routes them to your Amazon SNS topic to notify subscribed email addresses.

To create a rule that routes events to your Amazon SNS topic

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. In the navigation pane, under **Events**, choose **Rules**.
3. In the **Rules** section, choose **Create rule**.
4. Enter a name and description for the rule.
5. For **Define pattern**, do the following:

    a. Choose **Event Pattern**.

    b. For **Event matching pattern**, choose **Pre-defined by service**.

    c. For **Service provider**, choose **AWS**.

    d. For **Service Name**, choose **Auto Scaling**.

    e. For **Event type**, choose **Instance Refresh**.

    f. By default, the rule matches any Auto Scaling group in the Region. To make the rule match a specific Auto Scaling group, choose **Specific group name(s)** and select one or more Auto Scaling groups.

6. For **Select event bus**, choose **AWS default event bus**. When an AWS service in your account emits an event, it always goes to your account's default event bus.
7. For **Target**, choose **SNS topic**.
8. For **Topic**, select the Amazon SNS topic that you created.
9. For **Configure input**, choose the input for the email notification.
10. Choose **Create**.

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

The AWS shared responsibility model applies to data protection in Amazon EC2 Auto Scaling. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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). That way 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 SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- 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.

- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon EC2 Auto Scaling or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon EC2 Auto Scaling or other services 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.

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. To specify a key, use a launch template instead. For more information, see Creating a launch template for an Auto Scaling group (p. 26). 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 AWS KMS, see What is AWS Key Management Service?

Related topics

- Required CMK key policy for use with encrypted volumes (p. 261)
- Data protection in Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*
- Encryption by default 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

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

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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions 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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can
perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies.
Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify
a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific
resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to
indicate that the statement applies to all resources.

```
"Resource": "*"
```

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) in the *AWS General Reference*.

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`
- `DescribeInstanceRefreshes`
- `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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create 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*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys 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:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`
- `autoscaling:MetadataHttpTokens`
- `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, resources, and condition keys for Amazon EC2 Auto Scaling in the *IAM User Guide*.

Amazon EC2 Auto Scaling resource-based policies

Other AWS services, such as Amazon Simple Storage Service, support resource-based permissions policies. For example, you can attach a permissions policy to an S3 bucket to manage access permissions to that bucket.

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 see examples of identity-based policies based on tags, see Amazon EC2 Auto Scaling identity-based policy examples (p. 246).

To view an example policy that controls who can delete scaling policies based on the tags on the Auto Scaling group, see Control which scaling policies can be deleted (p. 250).

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. 243).

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. 158).

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. 259).

Learn more about IAM permission policies

Use the following topics to learn more about creating IAM permission policies to control who can or cannot use specific API actions.

- Amazon EC2 Auto Scaling
  - Actions, resources, and condition keys for Amazon EC2 Auto Scaling in the *IAM User Guide*
  - Amazon EC2 Auto Scaling identity-based policy examples (p. 246)

- Amazon EC2 launch templates
  - Actions, resources, and condition keys for Amazon EC2 in the *IAM User Guide*
  - Launch template support (p. 255)

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. 222).

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. 244).
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. 261).
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. 253).

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. 253) 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. 107).

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. 247)
- Predefined AWS managed policies (p. 247)
- Customer managed policy examples (p. 248)
- Required permissions to create a service-linked role (p. 253)
- Required permissions for the API (p. 254)

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. 238).

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

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 minimum and maximum capacity 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 Amazon EC2 Auto Scaling resources that specify 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. 259).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/qateam-*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        }
    ]
}
```

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. 239).

```
        "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. 239). 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 Launch template support (p. 255).

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. 243).

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 Amazon EC2 Auto Scaling and Amazon 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

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. 251).

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 (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** 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**.

The `iam:PassRole` permission is needed on the IAM user who creates or updates an Auto Scaling group using a launch template that specifies an instance profile, or who creates a launch configuration that specifies an instance profile. For an example policy, see Control which IAM roles can be passed (using PassRole) (p. 251).

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 Configuring advanced settings for your launch template (p. 30).

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"}}'
```

Create a launch configuration

When you create the launch configuration using the AWS Management Console, in the **Additional configuration** section, select the role from **IAM instance profile**. For more information, see Creating a launch configuration (p. 39).

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
```

Required CMK key policy for use with encrypted volumes

Amazon EC2 Auto Scaling supports service-linked roles (p. 243), a new type of IAM role that gives you a more secure and transparent way to delegate permissions to AWS services. Amazon EC2 Auto Scaling service-linked roles are predefined by Amazon EC2 Auto Scaling and include all the permissions that the service requires to call other AWS services on your behalf. The predefined permissions also include access to your AWS managed customer master keys (CMKs). However, they do not include access to your customer managed CMKs, allowing you to maintain full control over these keys.

Configuring key policies

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 CMKs 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.

If you specify a customer managed CMK for Amazon EBS encryption, you must give the appropriate service-linked role 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.

> **Note**
> 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.

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`.

For more information, 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. 243).
- For more information about Amazon EBS and KMS, Amazon EBS Encryption in the *Amazon EC2 User Guide for Linux Instances* and the AWS Key Management Service Developer Guide.

Example 1: 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 2: 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": "AllowCreationOfGrantForTheCMKinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

After you grant these permissions, everything should work as expected. For more information, see this forum post.

Editing key policies in the console

The examples in the previous 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.

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

You can create a VPC endpoint for the Amazon EC2 Auto Scaling service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). 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.

- **cn.com.amazonaws.*region*.autoscaling** — Creates an endpoint for the Amazon EC2 Auto Scaling API operations in the AWS China (Beijing) Region and AWS China (Ningxia) Region.

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 service 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 delete 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:DeleteScalingPolicy",
            "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.

Topics

Retrieving an error message from scaling activities

To retrieve an error message from the description of scaling activities, use the **describe-scaling-activities** command. You have a record of scaling activities that dates back 6 weeks. Scaling activities are ordered by start time, with the latest scaling activities listed first.

To see the scaling activities for a specific Auto Scaling group, use the following command.

```
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": [
        {
            "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e.  Status
 Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with token
 e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed with
 ABANDON Result",
            "Cause": "At 2021-01-11T00:35:52Z a user request created an AutoScalingGroup
 changing the desired capacity from 0 to 1.  At 2021-01-11T00:35:53Z an instance was
 started in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 1.",
            "StartTime": "2021-01-11T00:35:55.542Z",
            "EndTime": "2021-01-11T01:06:31Z",
            "StatusCode": "Cancelled",
            "StatusMessage": "Instance failed to complete user's Lifecycle Action:
 Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle
 Action Completed with ABANDON Result",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
```

```
       ...
    ]
}
```

For a description of the fields in the output, see Activity in the *Amazon EC2 Auto Scaling API Reference*.

To view scaling activities for a deleted Auto Scaling group, add the `--include-deleted-groups` option to the **describe-scaling-activities** command as follows.

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

The following is an example response, with a scaling activity for a deleted group.

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance.  Status Reason: Your Spot request
 price of 0.001 is lower than the minimum required Spot request fulfillment price of
 0.0031. Launching EC2 instance failed.",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup
 constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3.  At
 2021-01-13T20:47:27Z an instance was started in response to a difference between desired
 and actual capacity, increasing the capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than the minimum
 required Spot request fulfillment price of 0.0031. Launching EC2 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
        ...
    ]
}
```

The following tables list the types of error messages and provide links to the troubleshooting resources that you can use to troubleshoot issues.

Launch issues

Issue	Error message
Availability Zone	The requested Availability Zone is no longer supported. Please retry your request... (p. 273)
Block device mapping	Invalid device name upload. Launching EC2 instance failed. (p. 273)
Block device mapping	Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 274)
Block device mapping	EBS block device mappings not supported for instance-store AMIs. (p. 274)
Instance configuration	The requested configuration is currently not supported. (p. 271)

Issue	Error message
Instance type and Availability Zone	Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)... (p. 273)
Insufficient instance capacity in Availability Zone	We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed. (p. 275)
Insufficient instance capacity for a Spot request	There is no Spot capacity available that matches your request. Launching EC2 instance failed. (p. 275)
Key pair	The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 272)
Placement group	Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 274)
Quota limits	<number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 276)
Security group	The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 272)
Service-linked role	Client.InternalError: Client error on launch. (p. 274)
Spot price too low	Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735... (p. 273)

AMI issues

Issue	Error message
AMI ID	The AMI ID <ID of your AMI> does not exist. Launching EC2 instance failed. (p. 276)
AMI ID	AMI <AMI ID> is pending, and cannot be run. Launching EC2 instance failed. (p. 277)
AMI ID	Value (<ami ID>) for parameter virtualName is invalid. (p. 277)
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. 277)

Load balancer issues

Issue	Error message
Cannot find load balancer	Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 278)
Instances in VPC	EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 278)
No active load balancer	There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 278)

Issue	Error message
Security token	The security token included in the request is invalid. Validating load balancer configuration failed. (p. 279)

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 from scaling activities (p. 268).

When your EC2 instances fail to launch, you might get one or more of the following error messages:

Launch issues

- The requested configuration is currently not supported. (p. 271)
- The security group <name of the security group> does not exist. Launching EC2 instance failed. (p. 272)
- The key pair <key pair associated with your EC2 instance> does not exist. Launching EC2 instance failed. (p. 272)
- The requested Availability Zone is no longer supported. Please retry your request... (p. 273)
- Your requested instance type (<instance type>) is not supported in your requested Availability Zone (<instance Availability Zone>)... (p. 273)
- Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735... (p. 273)
- Invalid device name upload. Launching EC2 instance failed. (p. 273)
- Value (<name associated with the instance storage device>) for parameter virtualName is invalid... (p. 274)
- EBS block device mappings not supported for instance-store AMIs. (p. 274)
- Placement groups may not be used with instances of type 'm1.large'. Launching EC2 instance failed. (p. 274)
- Client.InternalError: Client error on launch. (p. 274)
- We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed. (p. 275)
- There is no Spot capacity available that matches your request. Launching EC2 instance failed. (p. 275)
- <number of instances> instance(s) are already running. Launching EC2 instance failed. (p. 276)

The requested configuration is currently not supported.

- **Cause**: Some options in your launch template or launch configuration might not be compatible with the instance type, or the instance configuration might not be supported in your requested AWS Region or Availability Zones.
- **Solution**:

Amazon EC2 Auto Scaling User Guide
The security group <name of the security group>
does not exist. Launching EC2 instance failed.

Try a different instance configuration. To search for an instance type that meets your requirements, see Finding an Amazon EC2 instance type in the *Amazon EC2 User Guide for Linux Instances*.

For further guidance to resolve this issue, check the following:

- Ensure that you have chosen an AMI that is supported by your instance type. For example, if the instance type uses an Arm-based AWS Graviton processor instead of an Intel Xeon processor, you need an Arm-compatible AMI.

- Test that the instance type is available in your requested Region and Availability Zones. The newest generation instance types might not yet be available in a given Region or Availability Zone. The older generation instance types might not be available in newer Regions and Availability Zones. To search for instance types offered by location (Region or Availability Zone), use the **describe-instance-type-offerings** command. For more information, see Finding an Amazon EC2 instance type in the *Amazon EC2 User Guide for Linux Instances*.

- If you use Dedicated Instances or Dedicated Hosts, ensure that you have chosen an instance type that's supported as a Dedicated Instance or Dedicated Host.

The security group <name of the security group> does not exist. Launching EC2 instance failed.

- **Cause**: The security group specified in your launch template or 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 template or launch configuration.
  4. Update your Auto Scaling group with the new launch template or 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 template or launch configuration.
  4. Update your Auto Scaling group with the new launch template or launch configuration using the **update-auto-scaling-group** command.

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 you chose 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.

Your Spot request price of 0.015 is lower than the minimum required Spot request fulfillment price of 0.0735...

- **Cause**: The Spot maximum price in your request is lower than the Spot price for the instance type that you selected.
- **Solution**: Submit a new request with a higher Spot maximum price (possibly the On-Demand price). Previously, the Spot price you paid was based on bidding. Today, you pay the current Spot price. By setting your maximum price higher, it gives the Amazon EC2 Spot service a better chance of launching and maintaining your required amount of capacity.

Invalid device name upload. Launching EC2 instance failed.

- **Cause**: The block device mappings in your launch template or 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 template or launch configuration using the device name listed in the volume description.
  3. Update your Auto Scaling group with the new launch template or 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 template or launch configuration by specifying the device name in the `virtualName` parameter. For information about the device name format, see Device naming on Linux instances in the *Amazon EC2 User Guide for Linux Instances*.
  2. Update your Auto Scaling group with the new launch template or 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 template or launch configuration are not supported on your instance.
- **Solution**:
  1. Create a new launch template or 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 template or 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 template or launch configuration with the supported instance type.
  4. Update your Auto Scaling group with a new placement group, launch template, or launch configuration using the **update-auto-scaling-group** command.

Client.InternalError: Client error on launch.

- **Problem**: Amazon EC2 Auto Scaling tries 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. 261).
- **Cause 1**: You need a key policy that gives permission to use the CMK to the proper service-linked role.
- **Solution 1**: Allow the service-linked role to use the CMK as follows:

Amazon EC2 Auto Scaling User Guide
We currently do not have sufficient <instance
type> capacity in the Availability Zone you
requested... Launching EC2 instance failed.

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.

For an example of a key policy that lets the service-linked role use the CMK, see Example 1: Key policy sections that allow access to the CMK (p. 262).

- **Cause 2**: If the CMK and Auto Scaling group are in *different* AWS accounts, you need to configure cross-account access to the CMK in order to give permission to use the CMK to the proper service-linked role.

- **Solution 2**: Allow the service-linked role in the external account to use the CMK in the local account as follows:

1. Update the key policy on the CMK to allow the Auto Scaling group account access to the CMK.

2. Define an IAM user or role in the Auto Scaling group account that can create a grant.

3. Determine which service-linked role to use for this Auto Scaling group.

4. Create a grant to the CMK with the proper service-linked role as the grantee principal.

5. Update the Auto Scaling group to use the service-linked role.

For more information, see Example 2: Key policy sections that allow cross-account access to the CMK (p. 263).

- **Solution 3**: 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 Solution 1.

We currently do not have sufficient <instance type> capacity in the Availability Zone you requested... Launching EC2 instance failed.

- **Error message**: We currently do not have sufficient <instance type> capacity in the Availability Zone you requested (<requested Availability Zone>). Our system will be working on provisioning additional capacity. You can currently get <instance type> capacity by not specifying an Availability Zone in your request or choosing <list of Availability Zones that currently supports the instance type>. Launching EC2 instance failed.

- **Cause**: At this time, Amazon EC2 cannot support your instance type in your requested Availability Zone.

- **Solution**:

To resolve the issue, try the following:

- Wait a few minutes and then submit your request again; capacity can shift frequently.

- Submit a new request following the recommendations in the error message.

- Submit a new request with a reduced number of instances (which you can increase at a later stage).

There is no Spot capacity available that matches your request. Launching EC2 instance failed.

- **Cause**: At this time, there isn't enough spare capacity to fulfill your request for Spot Instances.

- **Solution**:

Amazon EC2 Auto Scaling User Guide
<number of instances> instance(s) are already
running. Launching EC2 instance failed.

To resolve the issue, try the following:

- Wait a few minutes; capacity can shift frequently. The Spot request continues to automatically make the launch request until capacity becomes available. When capacity becomes available, the Amazon EC2 Spot service fulfills the Spot request.
- Follow the best practice of using a diverse set of instance types so that you are not reliant on one specific instance type. For more information, including a list of best practices for using Spot Instances successfully, see Auto Scaling groups with multiple instance types and purchase options (p. 53).
- Submit a new request with a reduced number of instances (which you can increase at a later stage).

<number of instances> instance(s) are already running. Launching EC2 instance failed.

- **Cause**: You have reached the limit on the number of instances that you can launch in a Region. When you create your AWS account, we set default limits on the number of instances you can run on a per-Region basis.
- **Solution**:

  To resolve the issue, try the following:

  - If your current limits aren't adequate for your needs, you can request a quota increase on a per-Region basis. For more information, see Amazon EC2 service quotas in the *Amazon EC2 User Guide for Linux Instances*.
  - Submit a new request with a reduced number of instances (which you can increase at a later stage).

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 from scaling activities (p. 268).

When your EC2 instances fail to launch due to issues with your AMI, you might get one or more of the following error messages.

AMI issues

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 template or launch configuration.

Amazon EC2 Auto Scaling User Guide
AMI <AMI ID> is pending, and cannot
be run. Launching EC2 instance failed.

- **Solution**:

  1. Create a new launch template or launch configuration using a valid AMI.

  2. Update your Auto Scaling group with the new launch template or 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 template or 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 template or 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 template or 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 template or launch configuration does not match the image architecture.
- **Solution**:

  1. Create a new launch template or 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 template or 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 from scaling activities (p. 268).

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.

Load balancer issues

- Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed. (p. 278)
- There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed. (p. 278)
- EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed. (p. 278)
- EC2 instance <instance ID> is in VPC. Updating load balancer configuration failed. (p. 279)
- The security token included in the request is invalid. Validating load balancer configuration failed. (p. 279)

Cannot find Load Balancer <your launch environment>. Validating load balancer configuration failed.

- **Cause 1**: The load balancer has been deleted.
- **Solution 1**:
  1. Check to see if your load balancer still exists. You can use the **describe-load-balancers** command.
  2. If you see your load balancer listed in the response, see **Cause 2**.
  3. If you do not see your load balancer listed in the response, you can either create a new load balancer and then create a new Auto Scaling group or you can create a new Auto Scaling group without the load balancer.
- **Cause 2**: The load balancer name was not specified in the right order when creating the Auto Scaling group.
- **Solution 2**: Create a new Auto Scaling group and specify the load balancer name at the end.

There is no ACTIVE Load Balancer named <load balancer name>. Updating load balancer configuration failed.

- **Cause**: The specified load balancer might have been deleted.
- **Solution**: You can either create a new load balancer and then create a new Auto Scaling group or create a new Auto Scaling group without the load balancer.

EC2 instance <instance ID> is not in VPC. Updating load balancer configuration failed.

- **Cause**: The specified instance does not exist in the VPC.
- **Solution**: You can either delete your load balancer associated with the instance or create a new Auto Scaling group.

Amazon EC2 Auto Scaling User Guide
EC2 instance <instance ID> is in VPC.
Updating load balancer configuration failed.

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 Amazon EC2 Auto Scaling: Launch templates

Use the following information to help you diagnose and fix common issues that you might encounter when using a launch template with your Auto Scaling group.

You are not authorized to perform this operation

Problem: When you try to specify a launch template, you get the "You are not authorized to perform this operation" error.

Cause 1: If you are attempting to use a launch template, and the IAM credentials that you are using do not have sufficient permissions, you receive an error that you're not authorized to use the launch template.

Solution 1: Verify that the IAM user or role that you are using to make the request has the correct permissions. For information about the permissions necessary to work with launch templates, see Launch template support (p. 255).

Cause 2: If you are attempting to use a launch template that specifies an instance profile, you must have permission to pass the IAM role that is associated with the instance profile.

Solution 2: Verify that the IAM user or role that you are using to make the request has the correct permissions to pass the specified role to the Amazon EC2 Auto Scaling service. For more information, see IAM role for applications that run on Amazon EC2 instances (p. 259). For further troubleshooting topics related to instance profiles, see Troubleshooting Amazon EC2 and IAM in the *IAM User Guide*.

Troubleshooting Amazon EC2 Auto Scaling: Health checks

This page provides information about your EC2 instances that terminate due to a health check. It describes potential causes, and the steps that you can take to resolve the issues.

To retrieve an error message, see Retrieving an error message from scaling activities (p. 268).

Amazon EC2 Auto Scaling User Guide
An instance was taken out of service in
response to an EC2 instance status check failure

Health check issues

> **Note**
> You can be notified when Amazon EC2 Auto Scaling terminates the instances in your Auto Scaling group, including when the cause of instance termination is not the result of a scaling activity. For more information, see Getting Amazon SNS notifications when your Auto Scaling group scales (p. 224).
> The sections that follow describe the most common health check errors and causes that you'll encounter. If you have a different issue, see the following AWS Knowledge Center articles for additional troubleshooting help:
>
> - Why did Amazon EC2 Auto Scaling terminate an instance?
> - Why didn't Amazon EC2 Auto Scaling terminate an unhealthy instance?

An instance was taken out of service in response to an EC2 instance status check failure

Problem: Instances in the Auto Scaling group fail the Amazon EC2 status checks.

Cause 1: If there are issues that cause Amazon EC2 to consider the instances in your Auto Scaling group impaired, Amazon EC2 Auto Scaling automatically replaces the impaired instances as part of its health check. Status checks are built into Amazon EC2, so they cannot be disabled or deleted. When an instance status check fails, you typically must address the problem yourself by making instance configuration changes until your application is no longer exhibiting any problems.

Solution 1: To address this issue, follow these steps:

1. Manually create an Amazon EC2 instance that is not part of the Auto Scaling group and investigate the problem. For general help with investigating impaired instances, see Troubleshoot instances with failed status checks in the *Amazon EC2 User Guide for Linux Instances* and Troubleshooting Windows Instances in the *Amazon EC2 User Guide for Windows Instances*.
2. After you confirm that everything works, deploy a new, error-free instance configuration to the Auto Scaling group.
3. Delete the instance that you created to avoid ongoing charges to your AWS account.

Cause 2: There is a mismatch between the health check grace period and the instance startup time.

Solution 2: Edit the health check grace period for your Auto Scaling group to an appropriate time period for your application. Instances launched in an Auto Scaling group require sufficient warm-up time (grace period) to prevent early termination due to a health check replacement. For more information, see Health check grace period (p. 206).

An instance was taken out of service in response to an EC2 scheduled reboot

Problem: Instances are replaced immediately when a scheduled event indicates a problem with the instance.

Amazon EC2 Auto Scaling User Guide
An instance was taken out of service in
response to an EC2 health check that
indicated it had been terminated or stopped

Cause: Amazon EC2 Auto Scaling replaces instances with a future scheduled maintenance or retirement event before the scheduled time.

Solution: These events do not occur frequently. If you do not want instances to be replaced due to a scheduled event, you can suspend the health check process for an Auto Scaling group. For more information, see Suspending and resuming scaling processes (p. 199).

Alternatively, if you need something to happen on the instance that is terminating, or on the instance that is starting up, you can use lifecycle hooks. These hooks allow you to perform a custom action as Amazon EC2 Auto Scaling launches or terminates instances. For more information, see Amazon EC2 Auto Scaling lifecycle hooks (p. 158).

An instance was taken out of service in response to an EC2 health check that indicated it had been terminated or stopped

Problem: Instances in the Auto Scaling group that have been stopped, rebooted, or terminated are replaced.

Cause 1: A user manually stopped, rebooted, or terminated the instance.

Solution 1: If a health check fails because a user manually stopped, rebooted, or terminated the instance, this is due to how Amazon EC2 Auto Scaling health checks work. The instance must be healthy and reachable. If you need to reboot the instances in your Auto Scaling group, we recommend that you put the instances on standby first. For more information, see Temporarily removing instances from your Auto Scaling group (p. 195).

Note that when you terminate instances manually, termination lifecycle hooks and Elastic Load Balancing deregistration (and connection draining) must be completed before the instance is actually terminated.

Cause 2: Amazon EC2 Auto Scaling attempts to replace Spot Instances after the Amazon EC2 Spot service interrupts the instances, because the Spot price increases above your maximum price or capacity is no longer available.

Solution 2: There is no guarantee that a Spot Instance exists to fulfill the request at any given point in time. However, you can try the following:

- Use a higher Spot maximum price (possibly the On-Demand price). By setting your maximum price higher, it gives the Amazon EC2 Spot service a better chance of launching and maintaining your required amount of capacity.

- Increase the number of different capacity pools that you can launch instances from by running multiple instance types in multiple Availability Zones. For more information, see Auto Scaling groups with multiple instance types and purchase options (p. 53).

- If you use multiple instance types, consider enabling the Capacity Rebalancing feature. This is useful if you want the Amazon EC2 Spot service to attempt to launch a new Spot Instance before a running instance is terminated. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing (p. 208).

Amazon EC2 Auto Scaling User Guide
An instance was taken out of service in
response to an ELB system health check failure

An instance was taken out of service in response to an ELB system health check failure

Problem: Instances in your Auto Scaling group might pass the EC2 status checks. But they might fail the Elastic Load Balancing health checks for the target groups or Classic Load Balancers with which the Auto Scaling group is registered.

Cause: If your Auto Scaling group relies on health checks provided by Elastic Load Balancing, Amazon EC2 Auto Scaling determines the health status of your instances by checking the results of both the EC2 status checks and the Elastic Load Balancing health checks. The load balancer performs health checks by sending a request to each instance and waiting for the correct response, or by establishing a connection with the instance. An instance might fail the Elastic Load Balancing health check because an application running on the instance has issues that cause the load balancer to consider the instance out of service. For more information, see Adding Elastic Load Balancing health checks to an Auto Scaling group (p. 91).

Solution 1: To pass the Elastic Load Balancing health checks:

- Make note of the success codes that the load balancer is expecting, and verify that your application is configured correctly to return these codes on success.
- Verify that the security groups for your load balancer and Auto Scaling group are correctly configured.
- Verify that the health check settings of your target groups are correctly configured. You define health check settings for your load balancer per target group.
- Set the health check grace period for your Auto Scaling group to a long enough time period to support the number of consecutive successful health checks required before Elastic Load Balancing considers a newly launched instance healthy.
- Consider adding a launch lifecycle hook to the Auto Scaling group to ensure that the applications on the instances are ready to accept traffic before they are registered to the load balancer at the end of the lifecycle hook.
- Verify that the load balancer is configured in the same Availability Zones as your Auto Scaling group.

For more information, see the following topics:

- 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*
- Health checks for your target groups in the *User Guide for Gateway Load Balancers*
- Configure health checks for your Classic Load Balancer in the *User Guide for Classic Load Balancers*

Solution 2: Update the Auto Scaling group to disable Elastic Load Balancing health checks.

Amazon EC2 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 webpage 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. 284)	You can now access example templates for lifecycle hooks from GitHub. For more information, see Amazon EC2 Auto Scaling lifecycle hooks in the *Amazon EC2 Auto Scaling User Guide*.	April 9, 2021
Support for warm pools (p. 284)	You can now balance performance (minimize cold starts) and cost (stop over-provisioning instance capacity) for applications with long first boot times by adding warm pools to Auto Scaling groups. For more information, see Warm pools for Amazon EC2 Auto Scaling in the *Amazon EC2 Auto Scaling User Guide*.	April 8, 2021
Support for checkpoints (p. 284)	You can now add checkpoints to an instance refresh to replace instances in phases and perform verifications on your instances at specific points. For more information, see Adding checkpoints to an instance refresh in the *Amazon EC2 Auto Scaling User Guide*.	March 18, 2021
Guide changes (p. 284)	Improved documentation for using EventBridge with Amazon EC2 Auto Scaling events and lifecycle hooks. For more information, see Using Amazon EC2 Auto Scaling with EventBridge and Tutorial: Configure a lifecycle hook that invokes a Lambda function in the *Amazon EC2 Auto Scaling User Guide*.	March 18, 2021
Support for local time zones (p. 284)	You can now create recurring scheduled actions in the local time zone by adding the `--time-zone` option to the **put-scheduled-update-group-**	March 9, 2021

action command. If your time zone observes Daylight Saving Time (DST), the recurring action automatically adjusts for Daylight Saving Time. For more information, see Scheduled scaling in the *Amazon EC2 Auto Scaling User Guide*.

Expands functionality for mixed instances policies (p. 284)	You can now prioritize instance types for your Spot capacity when you use a mixed instances policy. Amazon EC2 Auto Scaling attempts to fulfill priorities on a best-effort basis but optimizes for capacity first. For more information, see Auto Scaling groups with multiple instance types and purchase options in the *Amazon EC2 Auto Scaling User Guide*.	March 8, 2021
Scaling activities for deleted groups (p. 284)	You can now view scaling activities for deleted Auto Scaling groups by adding the `--include-deleted-groups` option to the **describe-scaling-activities** command. For more information, see Troubleshooting Amazon EC2 Auto Scaling in the *Amazon EC2 Auto Scaling User Guide*.	February 23, 2021
Console improvements (p. 284)	You can now create and attach an Application Load Balancer or Network Load Balancer from the Amazon EC2 Auto Scaling console. For more information, see Create and attach a new Application Load Balancer or Network Load Balancer (console) in the *Amazon EC2 Auto Scaling User Guide*.	November 24, 2020
Multiple network interfaces (p. 284)	You can now configure a launch template for an Auto Scaling group that specifies multiple network interfaces. For more information, see Creating a launch template for an Auto Scaling group.	November 23, 2020

Multiple launch templates (p. 284)	Multiple launch templates can now be used with Auto Scaling groups. For more information, see Specifying a different launch template for an instance type in the *Amazon EC2 Auto Scaling User Guide*.	November 19, 2020
Gateway Load Balancers (p. 284)	Updated guide to show how to attach a Gateway Load Balancer to an Auto Scaling group to ensure that appliance instances launched by Amazon EC2 Auto Scaling are automatically registered and deregistered from the load balancer. For more information, see Elastic Load Balancing types and Attaching a load balancer to your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.	November 10, 2020
Maximum instance lifetime (p. 284)	You can now reduce the maximum instance lifetime to one day (86,400 seconds). For more information, see Replacing Auto Scaling instances based on maximum instance lifetime in the *Amazon EC2 Auto Scaling User Guide*.	November 9, 2020
Capacity Rebalancing (p. 284)	You can configure your Auto Scaling group to launch a replacement Spot Instance when Amazon EC2 emits a rebalance recommendation. For more information, see Amazon EC2 Auto Scaling Capacity Rebalancing in the *Amazon EC2 Auto Scaling User Guide*.	November 4, 2020
Instance metadata service version 2 (p. 284)	You can require the use of Instance Metadata Service Version 2, which is a session-oriented method for requesting instance metadata, when using launch configurations. For more information, see Configuring the instance metadata options in the *Amazon EC2 Auto Scaling User Guide*.	July 28, 2020

Guide changes (p. 284)	Various improvements and new console procedures in the Controlling which Auto Scaling instances terminate during scale in, Monitoring your Auto Scaling instances and groups, Launch templates, and Launch configurations sections of the *Amazon EC2 Auto Scaling User Guide*.	July 28, 2020
Instance refresh (p. 284)	Start an instance refresh to update all instances in your Auto Scaling group when you make a configuration change. For more information, see Replacing Auto Scaling instances based on an instance refresh in the *Amazon EC2 Auto Scaling User Guide*.	June 16, 2020
Guide changes (p. 284)	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. 284)	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. 284)	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. 284)	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. 284)	Various improvements and corrections in the How Amazon 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*.	February 10, 2020
Guide changes (p. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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. 284)	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 topic for scaling based on Amazon SQS (p. 284)	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

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

Feature	Description	Release date
Monitoring improvements	Auto Scaling group metrics no longer require that you enable detailed monitoring. You can now enable group metrics collection and view metrics graphs from the **Monitoring** tab in the console. For more information, see Monitoring your Auto Scaling groups and instances using Amazon CloudWatch.	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 EventBridge when it executes lifecycle hooks. For more information, see Getting EventBridge 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
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

Feature	Description	Release date
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
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

Feature	Description	Release date
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
Support for IAM	Added support for IAM. For more information, see Controlling access to your Amazon EC2 Auto Scaling resources.	2 December 2010