# 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.


   An illustration of a basic Auto Scaling group.
  

For more information about the benefits of Amazon EC2 Auto Scaling, see [Benefits of Auto Scaling](https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html).

**Auto Scaling Components**

The following table describes the key components of Amazon EC2 Auto Scaling.

|  |  |
| --- | --- |
| A graphic representing an Auto Scaling group. | **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](https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html). |
| A graphic representing a launch configuration. | **Configuration templates**  Your group uses a *launch template* or a *launch configuration* as a configuration template for its EC2 instances. You can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances. For more information, see [Launch Templates](https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchTemplates.html) and [Launch Configurations](https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchConfiguration.html). |
| A graphic representing scaling options. | **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](https://docs.aws.amazon.com/autoscaling/ec2/userguide/scaling_plan.html#scaling_typesof). |

## PCI DSS Compliance

Auto Scaling supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).

## Related 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.

To automatically distribute incoming application traffic across multiple instances in your Auto Scaling group, use Elastic Load Balancing.

To monitor basic statistics for your instances and Amazon EBS volumes, use Amazon CloudWatch.

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..

# Benefits of Auto Scaling

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

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

## Example: Distributing Instances Across Availability Zones

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.

### 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.


      A typical Auto Scaling group spanning two Availability
       Zones.
     

### Rebalancing Activities

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

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

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

Because Amazon EC2 Auto Scaling attempts to launch new instances before terminating the old ones, being at or near the specified maximum capacity could impede or completely halt rebalancing activities. To avoid this problem, the system can temporarily exceed the specified maximum capacity of a group by a 10 percent margin (or by a 1-instance margin, whichever is greater) during a rebalancing activity. The margin is extended only if the group is at or near maximum capacity and needs rebalancing, either because of user-requested rezoning or to compensate for zone availability issues. The extension lasts only as long as needed to rebalance the group typically a few minutes.

# Auto Scaling Lifecycle

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

The following illustration shows the transitions between instance states in the Amazon EC2 Auto Scaling lifecycle.


    The lifecycle of instances within an Auto Scaling group.
   

## 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 anautoscaling: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](https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html).

## 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.

# Launch Templates

A launch template is similar to a [launch configuration](https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchConfiguration.html), 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 such as tags or network configurations, and allow the other parameters to be specified as part of another version of the same template.

We recommend that you use launch templates instead of launch configurations to ensure that you can use the latest features of Amazon EC2, such as T2 Unlimited instances.

# Launch Configurations

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. Alternatively, if you are using launch templates, you can specify a launch template instead of a launch configuration or an EC2 instance..

# Creating a Launch Configuration Using an EC2 Instance

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

**Tip**

You can [create an Auto Scaling group directly from an EC2 instance](https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-from-instance.html). When you use this feature, Amazon EC2 Auto Scaling automatically creates a launch configuration for you as well.

If the specified instance has properties that are not currently supported by launch configurations, the instances launched by the Auto Scaling group might not be identical to the original EC2 instance.

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

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

# Auto Scaling Groups

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, if you specify instance types for which you have matching Reserved Instances, your discounted rate of the regular On-Demand Instance pricing also applies. The only difference between On-Demand Instances and Reserved Instances is that you must purchase the Reserved Instances in advance. All of these factors combined help you to optimize your cost savings for Amazon EC2 instances, while making sure that you obtain the desired scale and performance for your application.

## Allocation Strategies

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

### On-Demand Instances

The allocation strategy for On-Demand Instances is prioritized. The Auto Scaling group 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, 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. If you have unused Reserved Instances for c4.large, you can set the priority of your instance types to give the highest priority to your Reserved Instances by specifying c4.large as the highest priority instance type. When a c4.large instance launches, you receive the Reserved Instance pricing.

### Spot Instances

The allocation strategy for Spot Instances is lowest-price. Spot Instances are distributed across the number of Spot Instance pools that you specify, and come from the pools with the lowest price per unit at the time of fulfillment.

A Spot Instance pool is a set of unused EC2 instances with the same instance type, operating system, Availability Zone, and network platform. The number of instance types and Availability Zones (both of which are specified in the group) directly contributes to how many Spot pools Amazon EC2 Auto Scaling can draw from. For example, if you specify 4 instance types and 4 Availability Zones, your Auto Scaling group can potentially fulfill Spot capacity from as many as 16 different Spot 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 (the 2 least expensive Spot pools per Availability Zone).

## Controlling the Proportion of On-Demand Instances

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

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

**Example: Scaling behavior**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instances Distribution** | **Total Number of Running Instances Across Purchase Options** | | | |
|  | **10** | **20** | **30** | **40** |
| **Example 1** |  |  |  |  |
| On-Demand base: 10 | 10 | 10 | 10 | 10 |
| On-Demand percentage above base: 50% | 0 | 5 | 10 | 15 |
| Spot percentage: 50% | 0 | 5 | 10 | 15 |
| **Example 2** |  |  |  |  |
| On-Demand base: 0 | 0 | 0 | 0 | 0 |
| On-Demand percentage above base: 0% | 0 | 0 | 0 | 0 |
| Spot percentage: 100% | 10 | 20 | 30 | 40 |
| **Example 3** |  |  |  |  |
| On-Demand base: 0 | 0 | 0 | 0 | 0 |
| On-Demand percentage above base: 60% | 6 | 12 | 18 | 24 |
| Spot percentage: 40% | 4 | 8 | 12 | 16 |
| **Example 4** |  |  |  |  |
| On-Demand base: 0 | 0 | 0 | 0 | 0 |
| On-Demand percentage above base: 100% | 10 | 20 | 30 | 40 |
| Spot percentage: 0% | 0 | 0 | 0 | 0 |
| **Example 5** |  |  |  |  |
| On-Demand base: 12 | 10 | 12 | 12 | 12 |
| On-Demand percentage above base: 0% | 0 | 0 | 0 | 0 |
| Spot percentage: 100% | 0 | 8 | 18 | 28 |

## Best Practices for Spot Instances

Before you create your Auto Scaling group to request Spot Instances, review [Spot Best Practices](https://aws.amazon.com/ec2/spot/getting-started/#Spot_Best_Practices). Use these best practices when you plan your request so that you can provision the type of instances 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 market price for the Spot Instances that you launch. If the Spot market price is within your maximum price, whether your request is fulfilled depends on availability.
* Create your Auto Scaling group with multiple instance types. The minimum is two. Because prices fluctuate independently for each instance type in an Availability Zone, you can often get more compute capacity for the same price when you have instance type flexibility.
* Similarly, don't limit yourself to only the most popular instance types. Because prices adjust based on long-term demand, popular instance types (such as recently launched instance families), tend to have more price adjustments. Picking older-generation instance types that are less popular tends to result in lower costs and fewer interruptions.
* If you run a web service, specify a high number of Spot pools, for example, N=10, to reduce the impact of Spot Instance interruptions if a pool in one of the Availability Zones becomes temporarily unavailable. If you run batch processing or other non-mission critical applications, you can specify a lower number of Spot pools, for example, N=2. This helps to ensure that you provision Spot Instances from only the very lowest priced Spot pools available per Availability Zone.
* Use Spot Instance interruption notices to monitor the status of your Spot Instances. For example, you can set up a rule in Amazon CloudWatch Events that automatically sends the EC2 Spot two-minute warning to an Amazon SNS topic, an AWS Lambda function, or another target.

# Creating an Auto Scaling Group Using Launch template

# Creating an Auto Scaling Group Using Launch configuration

# Creating an Auto Scaling Group Using an EC2 Instance

# Creating an Auto Scaling Group Using the Amazon EC2 Launch Wizard

# Using a Load Balancer with an Auto Scaling Group

The purpose of automatic scaling is to automatically increase the size of your Auto Scaling group when demand goes up and decrease it when demand goes down. As capacity is increased or decreased, the Amazon EC2 instances being added or removed must be registered or deregistered with a load balancer. This enables your application to automatically distribute incoming web traffic across such a dynamically changing number of instances.

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

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

You can also use an Elastic Load Balancing health check with your instances to make sure that traffic is routed only to the healthy instances.

## Elastic Load Balancing Types

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

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

Classic Load Balancer

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

Application Load Balancer

Routes and load balances at the application layer (HTTP/HTTPS), and supports path-based routing. An Application Load Balancer can route requests to ports on one or more registered targets, such as EC2 instances, in your virtual private cloud (VPC).

**Note**

The Application Load Balancer target groups must have a target type ofinstance.

Network Load Balancer

Routes and load balances at the transport layer (TCP/UDP Layer-4), based on address information extracted from the TCP packet header, not from packet content. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.

**Note**

The Network Load Balancer target groups must have a target type of instance.

## Attaching a Load Balancer to Your Auto Scaling Group

Amazon EC2 Auto Scaling integrates with Elastic Load Balancing to enable you to attach one or more load balancers to an existing Auto Scaling group. After you attach the load balancer, it automatically registers the instances in the group and distributes incoming traffic across the instances.

When you attach a load balancer, it enters the Adding state while registering the instances in the group. After all instances in the group are registered with the load balancer, it enters the Addedstate. After at least one registered instance passes the health checks, it enters the InServicestate. After the load balancer enters the InService state, Amazon EC2 Auto Scaling can terminate and replace any instances that are reported as unhealthy. If no registered instances pass the health checks (for example, due to a misconfigured health check), the load balancer doesn't enter the InService state. Amazon EC2 Auto Scaling doesn't terminate and replace the instances.

When you detach a load balancer, it enters the Removing state while deregistering the instances in the group. The instances remain running after they are deregistered. If connection draining is enabled, Elastic Load Balancing waits for in-flight requests to complete or for the maximum timeout to expire (whichever comes first) before deregistering the instances. By default, connection draining is enabled for Application Load Balancers but must be enabled for Classic Load Balancers. For more information, see [Connection Draining](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/config-conn-drain.html) in the User Guide for Classic Load Balancers.

# Adding Elastic Load Balancing Health Checks to an Auto Scaling Group

The default health checks for an Auto Scaling group are EC2 status checks only. If an instance fails these status checks, the Auto Scaling group considers the instance unhealthy and replaces it. For more information, see [Health Checks for Auto Scaling Instances](https://docs.aws.amazon.com/autoscaling/ec2/userguide/healthcheck.html).

If you attached one or more load balancers or target groups to your Auto Scaling group, the group does not, by default, consider an instance unhealthy and replace it if it fails the load balancer health checks.

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

If you configure the Auto Scaling group to use Elastic Load Balancing health checks, it considers the instance unhealthy if it fails either the EC2 status checks or the load balancer health checks. If you attach multiple load balancers to an Auto Scaling group, all of them must report that the instance is healthy in order for it to consider the instance healthy. If one load balancer reports an instance as unhealthy, the Auto Scaling group replaces the instance, even if other load balancers report it as healthy.

**Note**

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

# 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.

# 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.

## 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.

**Manual scaling**

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.

**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.

**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, 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 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.

**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) together to scale your Amazon EC2 capacity faster.

# 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.

# 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.

# Detach EC2 Instances from Your Auto Scaling Group

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

* Move an instance out of one Auto Scaling group and attach it to a different group. For more information, see [Attach EC2 Instances to Your Auto Scaling Group](https://docs.aws.amazon.com/autoscaling/ec2/userguide/attach-instance-asg.html).
* Test an Auto Scaling group by creating it using existing instances running your application, and then detach these instances from the Auto Scaling group when your tests are complete.

When you detach instances, you have the option of decrementing the desired capacity for the Auto Scaling group by the number of instances you are detaching. If you choose not to decrement the capacity, Amazon EC2 Auto Scaling launches new instances to replace the ones that you detached. 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](https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-suspend-resume-processes.html#process-types).

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.

# Dynamic Scaling for Amazon EC2 Auto Scaling

When you configure dynamic scaling, you must define how to scale in response to changing demand. For example,

## 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.

## Multiple Scaling Policies

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 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 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 EC2 instance could trigger the CloudWatch alarm for the CPUUtilization metric at the same time that the SQS queue triggers the alarm for the custom metric.

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 CPU utilization 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% of 8 = 2). The intention is to prevent Amazon EC2 Auto Scaling from removing too many instances.

# Target Tracking Scaling Policies for Amazon EC2 Auto Scaling

You can use the Amazon EC2 console to apply a target tracking scaling policy based on a predefined metric. Alternatively, you can use the Amazon EC2 Auto Scaling CLI or API to apply a scaling policy based on a predefined or customized metric. The following predefined metrics are available:

* ASGAverageCPUUtilization—Average CPU utilization of the Auto Scaling group.
* ASGAverageNetworkIn—Average number of bytes received on all network interfaces by the Auto Scaling group.
* ASGAverageNetworkOut—Average number of bytes sent out on all network interfaces by the Auto Scaling group.
* ALBRequestCountPerTarget—Number of requests completed per target in an Application Load Balancer target group.

## Instance Warmup

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. 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.

# Scaling Cooldowns for Amazon EC2 Auto Scaling

The cooldown period helps to ensure that your Auto Scaling group doesn't launch or terminate additional instances before the previous scaling activity takes effect.

Amazon EC2 Auto Scaling supports cooldown periods when using simple scaling policies, but not when using other scaling policies

When you manually scale your Auto Scaling group, the default is not to wait for the cooldown period, but you can override the default and honor the cooldown period. If an instance becomes unhealthy, the Auto Scaling group does not wait for the cooldown period to complete before replacing the unhealthy instance.

Amazon EC2 Auto Scaling supports both default cooldown periods and scaling-specific cooldown periods.

## Default Cooldowns

The default cooldown period is applied when you create your Auto Scaling group. Its default value is 300 seconds. This cooldown period automatically applies to any scaling activities for simple scaling policies, and you can optionally request to have it apply to your manual scaling activities.

One common use for scaling-specific cooldowns is with a scale-in policy—a policy that terminates instances based on a specific criteria or metric. Because this policy terminates instances, Amazon EC2 Auto Scaling needs less time to determine whether to terminate additional instances. The default cooldown period of 300 seconds is too long, in which case a scaling-specific cooldown period of 180 seconds for your scale-in policy can reduce costs by allowing the group to scale in faster.

## Cooldowns and Spot Instances

You can create Auto Scaling groups to use [Spot Instances](https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-launch-spot-instances.html) instead of On-Demand or Reserved Instances. The cooldown period begins when the bid for any Spot Instance is successful.

**Controlling Which Auto Scaling Instances Terminate During Scale In**

To specify which instances to terminate first during scale in, configure a termination policy for the Auto Scaling group.

Optionally, you can use instance 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.

## Default Termination Policy

This section describes the default termination policy used by an Auto Scaling group when a scale-in event occurs. The default termination policy is designed to help ensure that your instances span Availability Zones evenly for high availability. The default policy is kept generic and flexible to cover a range of scenarios.

With the default termination policy, the behavior of the Auto Scaling group is as follows:

1. Determine which Availability Zone(s) have the most instances, and at least one instance that is not protected from scale in.

If there are multiple unprotected instances to choose from in the Availability Zone(s) with the most instances, an instance is selected for termination based on the following criteria (applied in the order shown).

1. [For [Auto Scaling Groups with Multiple Instance Types and Purchase Options](https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-purchase-options.html) only]

Determine which instance to terminate so as to align the remaining instances to the allocation strategy for the On-Demand or Spot Instance that is terminating, your current selection of instance types, and distribution across your N lowest priced Spot pools. If there is one such instance, terminate it. Otherwise, apply the next condition.

1. [For Auto Scaling groups that use a launch template]

Determine whether any of the instances use the oldest launch template. If there is one such instance, terminate it. (Note that there is one exception: if the group originally used a launch configuration. Amazon EC2 Auto Scaling terminates instances that use a launch configuration before instances with the oldest launch template.)

1. [For Auto Scaling groups that use a launch configuration]

Determine whether any of the instances use the oldest launch configuration. If there is one such instance, terminate it.

1. After applying all of the criteria in 2 through 4, if there are multiple unprotected instances to terminate, determine which instances are closest to the next billing hour. If there is one such instance, terminate it. (Terminating the instance closest to the next billing hour helps you maximize the use of your instances that have an hourly charge.)

**Note**

With On-Demand Instances, you pay for compute capacity by per hour or per second depending on which instances you run. 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](https://aws.amazon.com/ec2/pricing/).

1. If there are multiple unprotected instances closest to the next billing hour, choose one of these instances at random.

## Customizing the Termination Policy

Amazon EC2 Auto Scaling supports the following custom termination policies:

* OldestInstance. Terminate the oldest instance in the group. This option is useful when you're upgrading the instances in the Auto Scaling group to a new EC2 instance type. You can gradually replace instances of the old type with instances of the new type.
* NewestInstance. Terminate the newest instance in the group. This policy is useful when you're testing a new launch configuration but don't want to keep it in production.
* OldestLaunchConfiguration. Terminate instances that have the oldest launch configuration. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
* ClosestToNextInstanceHour. Terminate instances that are closest to the next billing hour. This policy helps you maximize the use of your instances and manage your Amazon EC2 usage costs.
* Default. Terminate instances according to the default termination policy. This policy is useful when you have more than one scaling policy for the group.
* OldestLaunchTemplate. Terminate instances that have the oldest launch template. With this policy, instances that use the non-current launch template are terminated first, followed by instances that use the oldest version of the current launch template. This policy is useful when you're updating a group and phasing out the instances from a previous configuration.
* AllocationStrategy. Terminate instances in the Auto Scaling group to align the remaining instances to the allocation strategy for the type of instance that is terminating (either a Spot Instance or an On-Demand Instance). This policy is useful when your preferred instance types have changed. You can gradually rebalance the distribution of Spot Instances across your N lowest priced Spot pools. You can also gradually replace On-Demand Instances of a lower priority type with On-Demand Instances of a higher priority type.

## Instance Protection

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 protection settings are disabled.

Instance 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](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/terminating-instances.html#Using_ChangingDisableAPITermination) 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](https://docs.aws.amazon.com/autoscaling/ec2/userguide/healthcheck.html). To prevent Amazon EC2 Auto Scaling from terminating unhealthy instances, suspend the ReplaceUnhealthy process. For more information, see [Suspending and Resuming Scaling Processes](https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-suspend-resume-processes.html).
* Spot Instance interruptions. A Spot instance is terminated when capacity is no longer available or the Spot price exceeds your maximum price.

# Amazon EC2 Auto Scaling Lifecycle Hooks

## Considerations When Using Lifecycle Hooks

Adding lifecycle hooks to your Auto Scaling group gives you greater control over how instances launch and terminate. The following are things to consider when adding a lifecycle hook to your Auto Scaling group, to help ensure that the group continues to perform as expected.

### Keeping Instances in a Wait State

Instances can remain in a wait state for a finite period of time. The default is one hour (3600 seconds). You can adjust this time in the following ways:

* Set the heartbeat timeout for the lifecycle hook when you create the lifecycle hook. With the [**put-lifecycle-hook**](https://docs.aws.amazon.com/cli/latest/reference/autoscaling/put-lifecycle-hook.html) command, use the --heartbeat-timeout parameter. With thePutLifecycleHook operation, use the HeartbeatTimeout parameter.
* Continue to the next state if you finish before the timeout period ends, using the [**complete-lifecycle-action**](https://docs.aws.amazon.com/cli/latest/reference/autoscaling/complete-lifecycle-action.html) command or the CompleteLifecycleAction operation.
* Postpone the end of the timeout period by recording a heartbeat, using the [**record-lifecycle-action-heartbeat**](https://docs.aws.amazon.com/cli/latest/reference/autoscaling/record-lifecycle-action-heartbeat.html) 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.

### Cooldowns and Custom Actions

Consider an Auto Scaling group with a lifecycle hook that supports a custom action at instance launch. When the application experiences an increase in demand, the group launches instances to add capacity. Because there is a lifecycle hook, the instance is put into the Pending:Wait state, which means that it is not available to handle traffic yet. When the instance enters the wait state, scaling actions due to simple scaling policies are suspended. When the instance enters the InService state, the cooldown period starts. When the cooldown period expires, any suspended scaling actions resume.

### Health Check Grace Period

If you add a lifecycle hook, the health check grace period does not start until the lifecycle hook actions complete and the instance enters the InService state.

### Lifecycle Action Result

At the conclusion of a lifecycle hook, the result is either ABANDON or CONTINUE.

If the instance is launching, CONTINUE indicates that your actions were successful, and that the instance can be put into service. Otherwise, ABANDON indicates that your custom actions were unsuccessful, and that the instance can be terminated.

If the instance is terminating, both ABANDON and CONTINUE allow the instance to terminate. However, ABANDON stops any remaining actions, such as other lifecycle hooks, while CONTINUEallows any other lifecycle hooks to complete.

### Spot Instances

You can use lifecycle hooks with Spot Instances. However, a lifecycle hook does not prevent an instance from terminating in the event that capacity is no longer available. In addition, when a Spot Instance terminates, you must still complete the lifecycle action (using the **complete-lifecycle-action** command or the CompleteLifecycleAction operation).

# 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.

# Monitoring Your Auto Scaling Instances and Groups

You can use the following features to monitor your Auto Scaling instances and groups.

**Health checks**

Amazon EC2 Auto Scaling periodically performs health checks on the instances in your Auto Scaling group and identifies any instances that are unhealthy. You can configure Auto Scaling to determine the health status of an instance using Amazon EC2 status checks, Elastic Load Balancing health checks, or custom health checks

**CloudWatch metrics**

Amazon EC2 Auto Scaling publishes data points to Amazon CloudWatch about your Auto Scaling groups. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time series data, known as metrics.

**CloudWatch Events**

Amazon EC2 Auto Scaling can submit events to Amazon CloudWatch Events when your Auto Scaling groups launch or terminate instances, or when a lifecycle action occurs. This enables you to invoke a Lambda function when the event occurs.

**SNS notifications**

Amazon EC2 Auto Scaling can send Amazon SNS notifications when your Auto Scaling groups launch or terminate instances.

**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.

# Controlling Access to Your Amazon EC2 Auto Scaling Resources

By default, a brand new IAM user has no permissions to do anything. To grant permissions to call Amazon EC2 Auto Scaling actions, you attach an IAM policy to the IAM users or groups that require the permissions it grants.

# Troubleshooting Amazon EC2 Auto Scaling: EC2 Instance Launch Failures

Study this topic from documentation if needed

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/ts-as-instancelaunchfailure.html>