# Elastic Load Balancing

## Network Load Balancers

# Elastic Load Balancing: Network Load Balancers

# Table of Contents

# What is a Network Load Balancer?

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

Elastic Load Balancing supports the following load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. You can select the type of load balancer that best suits your needs. This guide discusses Network Load Balancers. For more information about the other load balancers, see the User Guide for Application Load Balancers, the User Guide for Gateway Load Balancers, and the User Guide for Classic Load Balancers.

## Network Load Balancer components

A *load balancer* serves as the single point of contact for clients. The load balancer distributes incoming traffic across multiple targets, such as Amazon EC2 instances. This increases the availability of your application. You add one or more listeners to your load balancer.

A *listener* checks for connection requests from clients, using the protocol and port that you configure, and forwards requests to a target group.

Each *target group* routes requests to one or more registered targets, such as EC2 instances, using the TCP protocol and the port number that you specify. You can register a target with multiple target groups. You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group that is specified in a listener rule for your load balancer.

For more information, see the following documentation:

- Load Balancers (p. 9)
- Listeners (p. 19)
- Target Groups (p. 30)

## Network Load Balancer overview

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

When you enable an Availability Zone for the load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. By default, each load balancer node distributes traffic across the registered targets in its Availability Zone only. If you enable cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. For more information, see Availability Zones (p. 10).

If you enable multiple Availability Zones for your load balancer and ensure that each target group has at least one target in each enabled Availability Zone, this increases the fault tolerance of your applications.

For example, if one or more target groups does not have a healthy target in an Availability Zone, we remove the IP address for the corresponding subnet from DNS, but the load balancer nodes in the other Availability Zones are still available to route traffic. If a client doesn't honor the time-to-live (TTL) and sends requests to the IP address after it is removed from DNS, the requests fail.

For TCP traffic, the load balancer selects a target using a flow hash algorithm based on the protocol, source IP address, source port, destination IP address, destination port, and TCP sequence number. The TCP connections from a client have different source ports and sequence numbers, and can be routed to different targets. Each individual TCP connection is routed to a single target for the life of the connection.

For UDP traffic, the load balancer selects a target using a flow hash algorithm based on the protocol, source IP address, source port, destination IP address, and destination port. A UDP flow has the same source and destination, so it is consistently routed to a single target throughout its lifetime. Different UDP flows have different source IP addresses and ports, so they can be routed to different targets.

Elastic Load Balancing creates a network interface for each Availability Zone you enable. Each load balancer node in the Availability Zone uses this network interface to get a static IP address. When you create an Internet-facing load balancer, you can optionally associate one Elastic IP address per subnet.

When you create a target group, you specify its target type, which determines whether you register targets by instance ID or IP address. If you register targets by instance ID, the source IP addresses of the clients are preserved and provided to your applications. If you register targets by IP address, the source IP addresses are the private IP addresses of the load balancer nodes.

You can add and remove targets from your load balancer as your needs change, without disrupting the overall flow of requests to your application. Elastic Load Balancing scales your load balancer as traffic to your application changes over time. Elastic Load Balancing can scale to the vast majority of workloads automatically.

You can configure health checks, which are used to monitor the health of the registered targets so that the load balancer can send requests only to the healthy targets.

For more information, see How Elastic Load Balancing works in the *Elastic Load Balancing User Guide*.

# Benefits of migrating from a Classic Load Balancer

Using a Network Load Balancer instead of a Classic Load Balancer has the following benefits:

- Ability to handle volatile workloads and scale to millions of requests per second.
- Support for static IP addresses for the load balancer. You can also assign one Elastic IP address per subnet enabled for the load balancer.
- Support for registering targets by IP address, including targets outside the VPC for the load balancer.
- Support for routing requests to multiple applications on a single EC2 instance. You can register each instance or IP address with the same target group using multiple ports.
- Support for containerized applications. Amazon Elastic Container Service (Amazon ECS) can select an unused port when scheduling a task and register the task with a target group using this port. This enables you to make efficient use of your clusters.
- Support for monitoring the health of each service independently, as health checks are defined at the target group level and many Amazon CloudWatch metrics are reported at the target group level. Attaching a target group to an Auto Scaling group enables you to scale each service dynamically based on demand.

For more information about the features supported by each load balancer type, see Product comparisons for Elastic Load Balancing.

# How to get started

To create a Network Load Balancer, try one of the following tutorials:

- Getting started with Network Load Balancers (p. 4)
- Tutorial: Create a Network Load Balancer using the AWS CLI (p. 7)

For demos of common load balancer configurations, see Elastic Load Balancing Demos.

# Pricing

For more information, see Network Load Balancer Pricing.

# Getting started with Network Load Balancers

This tutorial provides a hands-on introduction to Network Load Balancers through the AWS Management Console, a web-based interface. To create your first Network Load Balancer, complete the following steps.

**Tasks**



For demos of common load balancer configurations, see Elastic Load Balancing Demos.

## Before you begin

- Decide which Availability Zones you will use for your EC2 instances. Configure your virtual private cloud (VPC) with at least one public subnet in each of these Availability Zones. These public subnets are used to configure the load balancer. You can launch your EC2 instances in other subnets of these Availability Zones instead.
- Launch at least one EC2 instance in each Availability Zone. Ensure that the security groups for these instances allow TCP access from clients on the listener port and health check requests from your VPC. For more information, see Target security groups (p. 43).

## Step 1: Choose a load balancer type

Elastic Load Balancing supports different types of load balancers. For this tutorial, you create a Network Load Balancer.

**To create a Network Load Balancer**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar, choose a region for your load balancer. Be sure to choose the same region that you used for your EC2 instances.
3. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
4. Choose **Create Load Balancer**.
5. For **Network Load Balancer**, choose **Create**.

# Step 2: Configure your load balancer and listener

On the **Configure Load Balancer** page, complete the following procedure.

**To configure your load balancer and listener**

1. For **Name**, type a name for your load balancer.

   The name of your Network Load Balancer must be unique within your set of Application Load Balancers and Network Load Balancers for the region, can have a maximum of 32 characters, can contain only alphanumeric characters and hyphens, must not begin or end with a hyphen, and must not begin with "internal-".
2. For **Scheme** and **IP address type**, keep the default values.
3. For **Listeners**, keep the default, which is a listener that accepts TCP traffic on port 80.
4. For **Availability Zones**, select the VPC that you used for your EC2 instances. For each Availability Zone that you used to launch your EC2 instances, select the Availability Zone and then select one public subnet for that Availability Zone.

   By default, AWS assigns an IPv4 address to each load balancer node from the subnet for its Availability Zone. Alternatively, when you create an internet-facing load balancer, you can select an Elastic IP address for each Availability Zone. This provides your load balancer with static IP addresses.
5. Choose **Next: Configure Routing**.

# Step 3: Configure your target group

Create a target group, which is used in request routing. The rule for your listener routes requests to the registered targets in this target group. The load balancer checks the health of targets in this target group using the health check settings defined for the target group. On the **Configure Routing** page, complete the following procedure.

**To configure your target group**

1. For **Target group**, keep the default, **New target group**.
2. For **Name**, type a name for the new target group.
3. Keep **Protocol** as TCP, **Port** as 80, and **Target type** as instance.
4. For **Health checks**, keep the default protocol.
5. Choose **Next: Register Targets**.

# Step 4: Register targets with your target group

On the **Register Targets** page, complete the following procedure.

**To register targets with the target group**

1. For **Instances**, select one or more instances.
2. Keep the default port, 80, and choose **Add to registered**.
3. When you have finished selecting instances, choose **Next: Review**.

# Step 5: Create and test your load balancer

Before creating the load balancer, review your settings. After creating the load balancer, verify that it's sending traffic to your EC2 instances.

**To create and test your load balancer**

1. On the **Review** page, choose **Create**.
2. After you are notified that your load balancer was created successfully, choose **Close**.
3. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
4. Select the newly created target group.
5. Choose **Targets** and verify that your instances are ready. If the status of an instance is `initial`, it's probably because the instance is still in the process of being registered, or it has not passed the minimum number of health checks to be considered healthy. After the status of at least one instance is `healthy`, you can test your load balancer.
6. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
7. Select the newly created load balancer.
8. Choose **Description** and copy the DNS name of the load balancer (for example, my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com). Paste the DNS name into the address field of an internet-connected web browser. If everything is working, the browser displays the default page of your server.

# Step 6: Delete your load balancer (optional)

As soon as your load balancer becomes available, you are billed for each hour or partial hour that you keep it running. When you no longer need a load balancer, you can delete it. As soon as the load balancer is deleted, you stop incurring charges for it. Note that deleting a load balancer does not affect the targets registered with the load balancer. For example, your EC2 instances continue to run.

**To delete your load balancer**

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

# Tutorial: Create a Network Load Balancer using the AWS CLI

This tutorial provides a hands-on introduction to Network Load Balancers through the AWS CLI.

## Before you begin

- Install the AWS CLI or update to the current version of the AWS CLI if you are using a version that does not support Network Load Balancers. For more information, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
- Decide which Availability Zones you will use for your EC2 instances. Configure your virtual private cloud (VPC) with at least one public subnet in each of these Availability Zones.
- Launch at least one EC2 instance in each Availability Zone. Ensure that the security groups for these instances allow TCP access from clients on the listener port and health check requests from your VPC. For more information, see Target security groups (p. 43).

## Create your load balancer

To create your first load balancer, complete the following steps.

**To create a load balancer**

1. Use the create-load-balancer command to create a load balancer, specifying a public subnet for each Availability Zone in which you launched instances. You can specify only one subnet per Availability Zone.

```
aws elbv2 create-load-balancer --name my-load-balancer --type network --subnets
 subnet-0e3f5cac72EXAMPLE
```

   The output includes the Amazon Resource Name (ARN) of the load balancer, with the following format:

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-load-
balancer/1234567890123456
```

2. Use the create-target-group command to create a target group, specifying the same VPC that you used for your EC2 instances:

```
aws elbv2 create-target-group --name my-targets --protocol TCP --port 80 --vpc-id
 vpc-0598c7d356EXAMPLE
```

   The output includes the ARN of the target group, with this format:

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/1234567890123456
```

3. Use the register-targets command to register your instances with your target group:

```
aws elbv2 register-targets --target-group-arn targetgroup-arn --targets
 Id=i-1234567890abcdef0 Id=i-0abcdef1234567890
```

4. Use the create-listener command to create a listener for your load balancer with a default rule that forwards requests to your target group:

```
aws elbv2 create-listener --load-balancer-arn loadbalancer-arn --protocol TCP --port 80
  \
--default-actions Type=forward,TargetGroupArn=targetgroup-arn
```

   The output contains the ARN of the listener, with the following format:

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/net/my-load-
balancer/1234567890123456/1234567890123456
```

5. (Optional) You can verify the health of the registered targets for your target group using this describe-target-health command:

```
aws elbv2 describe-target-health --target-group-arn targetgroup-arn
```

# Specify an Elastic IP address for your load balancer

When you create a Network Load Balancer, you can specify one Elastic IP address per subnet using a subnet mapping.

```
aws elbv2 create-load-balancer --name my-load-balancer --type network \
--subnet-mappings SubnetId=subnet-0e3f5cac72EXAMPLE,AllocationId=eipalloc-12345678
```

# Delete your load balancer

When you no longer need your load balancer and target group, you can delete them as follows:

```
aws elbv2 delete-load-balancer --load-balancer-arn loadbalancer-arn
aws elbv2 delete-target-group --target-group-arn targetgroup-arn
```

# Network Load Balancers

A *load balancer* serves as the single point of contact for clients. Clients send requests to the load balancer, and the load balancer sends them to targets, such as EC2 instances, in one or more Availability Zones.

To configure your load balancer, you create target groups (p. 30), and then register targets with your target groups. Your load balancer is most effective if you ensure that each enabled Availability Zone has at least one registered target. You also create listeners (p. 19) to check for connection requests from clients and route requests from clients to the targets in your target groups.

Network Load Balancers support connections from clients over VPC peering, AWS managed VPN, AWS Direct Connect, and third-party VPN solutions.

**Contents**



## Load balancer state

A load balancer can be in one of the following states:

`provisioning`

The load balancer is being set up.

`active`

The load balancer is fully set up and ready to route traffic.

`failed`

The load balancer could not be set up.

## Load balancer attributes

The following are the load balancer attributes:

`deletion_protection.enabled`

Indicates whether deletion protection (p. 12) is enabled. The default is `false`.

`load_balancing.cross_zone.enabled`

Indicates whether cross-zone load balancing (p. 11) is enabled. The default is `false`.

# IP address type

You can set the types of IP addresses that clients can use with your internet-facing load balancer. The load balancer must have only TCP and TLS listeners. Clients must use IPv4 addresses with internal load balancers.

The following are the IP address types:

`ipv4`

Clients must connect to the load balancer using IPv4 addresses (for example, 192.0.2.1)

`dualstack`

Clients can connect to the load balancer using both IPv4 addresses (for example, 192.0.2.1) and IPv6 addresses (for example, 2001:0db8:85a3:0:0:8a2e:0370:7334).

When you enable dual-stack mode for the load balancer, Elastic Load Balancing provides an AAAA DNS record for the load balancer. Clients that communicate with the load balancer using IPv4 addresses resolve the A DNS record. Clients that communicate with the load balancer using IPv6 addresses resolve the AAAA DNS record.

The load balancer communicates with targets using IPv4 addresses, regardless of how the client communicates with the load balancer. Therefore, the targets do not need IPv6 addresses.

For more information, see Update the address type (p. 16).

# Availability Zones

You enable one or more Availability Zones for your load balancer when you create it. If you enable multiple Availability Zones for your load balancer, this increases the fault tolerance of your applications. You cannot disable Availability Zones for a Network Load Balancer after you create it, but you can enable additional Availability Zones.

When you enable an Availability Zone, you specify one subnet from that Availability Zone. Elastic Load Balancing creates a load balancer node in the Availability Zone and a network interface for the subnet (the description starts with "ELB net" and includes the name of the load balancer). Each load balancer node in the Availability Zone uses this network interface to get an IPv4 address. Note that you can view this network interface but you cannot modify it.

When you create an internet-facing load balancer, you can optionally specify one Elastic IP address per subnet. If you do not choose one of your own Elastic IP addresses, Elastic Load Balancing provides one Elastic IP address per subnet for you. These Elastic IP addresses provide your load balancer with static IP addresses that will not change during the life of the load balancer. You cannot change these Elastic IP addresses after you create the load balancer.

When you create an internal load balancer, you can optionally specify one private IP address per subnet. If you do not specify an IP address from the subnet, Elastic Load Balancing chooses one for you. These

private IP addresses provide your load balancer with static IP addresses that will not change during the life of the load balancer. You cannot change these private IP addresses after you create the load balancer.

### Requirements

- For internet-facing load balancers, the subnets that you specify must have at least 8 available IP addresses. For internal load balancers, this is only required if you let AWS select a private IPv4 address from the subnet.
- You can't specify a subnet in a constrained Availability Zone. The error message is "Load balancers with type 'network' are not supported in *az_name*". You can specify a subnet in another Availability Zone that is not constrained and use cross-zone load balancing to distribute traffic to targets in the constrained Availability Zone.
- You can't specify a subnet in a Local Zone.

After you enable an Availability Zone, the load balancer starts routing requests to the registered targets in that Availability Zone. Your load balancer is most effective if you ensure that each enabled Availability Zone has at least one registered target.

### To add Availability Zones using the 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. Select the load balancer.
4. On the **Description** tab, under **Basic Configuration**, choose **Edit subnets**.
5. To enable an Availability Zone, select the check box for that Availability Zone. If there is one subnet for that Availability Zone, it is selected. If there is more than one subnet for that Availability Zone, select one of the subnets. Note that you can select only one subnet per Availability Zone.

   For an internet-facing load balancer, you can select an Elastic IP address for each Availability Zone. For an internal load balancer, you can assign a private IP address from the IPv4 range of each subnet instead of letting Elastic Load Balancing assign one.
6. Choose **Save**.

### To add Availability Zones using the AWS CLI

Use the set-subnets command.

# Cross-zone load balancing

By default, each load balancer node distributes traffic across the registered targets in its Availability Zone only. If you enable cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. For more information, see Cross-zone load balancing in the *Elastic Load Balancing User Guide*.

### To enable cross-zone load balancing using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. Choose **Description**, **Edit attributes**.
5. On the **Edit load balancer attributes** page, select **Enable** for **Cross-Zone Load Balancing**, and choose **Save**.

**To enable cross-zone load balancing using the AWS CLI**

Use the modify-load-balancer-attributes command with the `load_balancing.cross_zone.enabled` attribute.

# Deletion protection

To prevent your load balancer from being deleted accidentally, you can enable deletion protection. By default, deletion protection is disabled for your load balancer.

If you enable deletion protection for your load balancer, you must disable it before you can delete the load balancer.

**To enable deletion protection using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. Choose **Description**, **Edit attributes**.
5. On the **Edit load balancer attributes** page, select **Enable** for **Delete Protection**, and choose **Save**.

**To disable deletion protection using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. Choose **Description**, **Edit attributes**.
5. On the **Edit load balancer attributes** page, clear **Enable delete protection** and choose **Save**.

**To enable or disable deletion protection using the AWS CLI**

Use the modify-load-balancer-attributes command with the `deletion_protection.enabled` attribute.

# Connection idle timeout

For each TCP request that a client makes through a Network Load Balancer, the state of that connection is tracked. If no data is sent through the connection by either the client or target for longer than the idle timeout, the connection is closed. If a client or a target sends data after the idle timeout period elapses, it receives a TCP RST packet to indicate that the connection is no longer valid.

Elastic Load Balancing sets the idle timeout value for TCP flows to 350 seconds. You cannot modify this value. Clients or targets can use TCP keepalive packets to reset the idle timeout.

While UDP is connectionless, the load balancer maintains UDP flow state based on the source and destination IP addresses and ports, ensuring that packets that belong to the same flow are consistently sent to the same target. After the idle timeout period elapses, the load balancer considers the incoming UDP packet as a new flow and routes it to a new target. Elastic Load Balancing sets the idle timeout value for UDP flows to 120 seconds.

EC2 instances must respond to a new request within 30 seconds in order to establish a return path.

# DNS name

Each Network Load Balancer receives a default Domain Name System (DNS) name with the following syntax: *name*-*id*.elb.*region*.amazonaws.com. For example, my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com.

If you'd prefer to use a DNS name that is easier to remember, you can create a custom domain name and associate it with the DNS name for your load balancer. When a client makes a request using this custom domain name, the DNS server resolves it to the DNS name for your load balancer.

First, register a domain name with an accredited domain name registrar. Next, use your DNS service, such as your domain registrar, to create a CNAME record to route requests to your load balancer. For more information, see the documentation for your DNS service. For example, you can use Amazon Route 53 as your DNS service. For more information, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.

The load balancer has one IP address per enabled Availability Zone. These are the addresses of the load balancer nodes. The DNS name of the load balancer resolves to these addresses. For example, suppose that the custom domain name for your load balancer is `example.networkloadbalancer.com`. Use the following **dig** or **nslookup** command to determine the IP addresses of the load balancer nodes.

**Linux or Mac**

```
$ dig +short example.networkloadbalancer.com
```

**Windows**

```
C:\> nslookup example.networkloadbalancer.com
```

The load balancer has DNS records for its load balancer nodes. You can use DNS names with the following syntax to determine the IP addresses of the load balancer nodes: *az*.*name*-*id*.elb.*region*.amazonaws.com.

**Linux or Mac**

```
$ dig +short us-east-2b.my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com
```

**Windows**

```
C:\> nslookup us-east-2b.my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com
```

# Create a Network Load Balancer

A load balancer takes requests from clients and distributes them across targets in a target group, such as EC2 instances.

Before you begin, ensure that the virtual private cloud (VPC) for your load balancer has at least one public subnet in each Availability Zone where you have targets.

To create a load balancer using the AWS CLI, see Tutorial: Create a Network Load Balancer using the AWS CLI (p. 7).

To create a load balancer using the AWS Management Console, complete the following tasks.

**Tasks**

# Step 1: Configure a load balancer and a listener

First, provide some basic configuration information for your load balancer, such as a name, a network, and one or more listeners. A listener is a process that checks for connection requests. It is configured with a protocol and a port for connections from clients to the load balancer. For more information about supported protocols and ports, see Listener configuration (p. 19).

**To configure your load balancer and listener**

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 **Create Load Balancer**.
4. For **Network Load Balancer**, choose **Create**.
5. For **Name**, type a name for your load balancer. For example, **`my-nlb`**.
6. For **Scheme**, an internet-facing load balancer routes requests from clients over the internet to targets. An internal load balancer routes requests to targets using private IP addresses.
7. For **IP address type**, choose **ipv4** if your clients use IPv4 addresses to communicate with the load balancer, or **dualstack** if your clients use both IPv4 and IPv6 addresses to communicate with the load balancer. If the load balancer is an internal load balancer, or if you will add a UDP or TCP_UDP listener, you must choose **ipv4**.
8. For **Listeners**, the default is a listener that accepts TCP traffic on port 80. You can keep the default listener settings, modify the protocol, or modify the port. Choose **Add** to add another listener.
9. For **Availability Zones**, select the VPC that you used for your EC2 instances. For each Availability Zone that you used to launch your EC2 instances, select an Availability Zone and then select the public subnet for that Availability Zone.

   By default, AWS assigns an IPv4 address to each load balancer node from the subnet for its Availability Zone. Alternatively, if you create an internet-facing load balancer, you can select an Elastic IP address for each Availability Zone. This provides your load balancer with static IP addresses. If you create an internal load balancer, you can assign a private IP address from the IPv4 range of each subnet instead of letting AWS assign one.
10. Choose **Next: Configure Routing**.

# Step 2: Configure a target group

You register targets, such as EC2 instances, with a target group. The target group that you configure in this step is used as the target group in the listener rule, which forwards requests to the target group. For more information, see Target groups for your Network Load Balancers (p. 30).

**To configure your target group**

1. For **Target group**, keep the default, **New target group**.
2. For **Name**, type a name for the target group.
3. For **Protocol**, choose a protocol as follows:

   - If the listener protocol is TCP, choose **TCP** or **TCP_UDP**.

- If the listener protocol is TLS, choose **TCP** or **TLS**.
- If the listener protocol is UDP, choose **UDP** or **TCP_UDP**.
- If the listener protocol is TCP_UDP, choose **TCP_UDP**.

4. (Optional) Set **Port** as needed.
5. For **Target type**, select `instance` to specify targets by instance ID or `ip` to specify targets by IP address.
6. For **Health checks**, keep the default health check settings.
7. Choose **Next: Register Targets**.

# Step 3: Register targets with the target group

You can register EC2 instances as targets in a target group.

### To register targets by instance ID

1. For **Instances**, select one or more instances.
2. Keep the default instance listener port or type a new one and choose **Add to registered**.
3. When you have finished registering instances, choose **Next: Review**.

### To register targets by IP address

1. For each IP address to register, do the following:
   a. For **Network**, if the IP address is from a subnet of the target group VPC, select the VPC. Otherwise, select **Other private IP address**.
   b. For **Availability Zone**, select an Availability Zone or **all**. This determines whether the target receives traffic from the load balancer nodes in the specified Availability Zone only or from all enabled Availability Zones. This field is not displayed if you are registering IP addresses from the VPC. In this case, the Availability Zone is automatically detected.
   c. For **IP**, type the address.
   d. For **Port**, type the port.
   e. Choose **Add to list**.
2. When you have finished adding IP addresses to the list, choose **Next: Review**.

# Step 4: Create the load balancer

After creating your load balancer, you can verify that your EC2 instances have passed the initial health check and then test that the load balancer is sending traffic to your EC2 instances. When you are finished with your load balancer, you can delete it. For more information, see Delete a Network Load Balancer (p. 17).

### To create the load balancer

1. On the **Review** page, choose **Create**.
2. After the load balancer is created, choose **Close**.
3. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
4. Select the newly created target group.
5. Choose **Targets** and verify that your instances are ready. If the status of an instance is `initial`, it's probably because the instance is still in the process of being registered, or it has not passed the

minimum number of health checks to be considered healthy. After the status of at least one instance is healthy, you can test your load balancer.

# IP address types for your Network Load Balancer

You can configure your Network Load Balancer so that clients can communicate with the load balancer using IPv4 addresses only, or using both IPv4 and IPv6 addresses. The load balancer communicates with targets using IPv4 addresses, regardless of how the client communicates with the load balancer. For more information, see IP address type (p. 10).

### IPv6 requirements

- An internet-facing load balancer with the `dualstack` IP address type. You can set the IP address type when you create the load balancer and update it at any time.
- The virtual private cloud (VPC) and subnets that you specify for the load balancer must have associated IPv6 CIDR blocks. For more information, see IPv6 addresses in the *Amazon EC2 User Guide*.
- The load balancer must have only TCP and TLS listeners.
- The route tables for the load balancer subnets must route IPv6 traffic.
- The network ACLs for the load balancer subnets must allow IPv6 traffic.

#### To set the IP address type at creation

Configure settings as described in Create a load balancer (p. 13).

### To update the IP address type using the 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. Select the load balancer.
4. Choose **Actions**, **Edit IP address type**.
5. For **IP address type**, choose **ipv4** to support IPv4 addresses only or **dualstack** to support both IPv4 and IPv6 addresses.
6. Choose **Save**.

#### To update the IP address type using the AWS CLI

Use the set-ip-address-type command.

# Tags for your Network Load Balancer

Tags help you to categorize your load balancers in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each load balancer. Tag keys must be unique for each load balancer. If you add a tag with a key that is already associated with the load balancer, it updates the value of that tag.

When you are finished with a tag, you can remove it from your load balancer.

### Restrictions

- Maximum number of tags per resource—50

- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case-sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- 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. Tags with this prefix do not count against your tags per resource limit.

### To update the tags for a load balancer using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer.
4. Choose **Tags**, **Add/Edit Tags**, and then do one or more of the following:
   a. To update a tag, edit the values of **Key** and **Value**.
   b. To add a new tag, choose **Create Tag**. For **Key** and **Value**, type values.
   c. To delete a tag, choose the delete icon (X) next to the tag.
5. When you have finished updating tags, choose **Save**.

### To update the tags for a load balancer using the AWS CLI

Use the add-tags and remove-tags commands.

# Delete a Network Load Balancer

As soon as your load balancer becomes available, you are billed for each hour or partial hour that you keep it running. When you no longer need the load balancer, you can delete it. As soon as the load balancer is deleted, you stop incurring charges for it.

You can't delete a load balancer if deletion protection is enabled. For more information, see Deletion protection (p. 12).

You can't delete a load balancer if it is in use by another service. For example, if the load balancer is associated with a VPC endpoint service, you must delete the endpoint service configuration before you can delete the associated load balancer.

Deleting a load balancer also deletes its listeners. Deleting a load balancer does not affect its registered targets. For example, your EC2 instances continue to run and are still registered to their target groups. To delete your target groups, see Delete a target group (p. 49).

### To delete a load balancer using the console

1. If you have a CNAME record for your domain that points to your load balancer, point it to a new location and wait for the DNS change to take effect before deleting your load balancer.
2. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
3. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
4. Select the load balancer.
5. Choose **Actions**, **Delete**.
6. When prompted for confirmation, choose **Yes, Delete**.

**To delete a load balancer using the AWS CLI**

Use the delete-load-balancer command.

# Listeners for your Network Load Balancers

Before you start using your Network Load Balancer, you must add one or more *listeners*. A listener is a process that checks for connection requests, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to the targets in one or more target groups.

For more information, see Request routing in the *Elastic Load Balancing User Guide*.

**Contents**



## Listener configuration

Listeners support the following protocols and ports:

- **Protocols**: TCP, TLS, UDP, TCP_UDP
- **Ports**: 1-65535

You can use a TLS listener to offload the work of encryption and decryption to your load balancer so that your applications can focus on their business logic. If the listener protocol is TLS, you must deploy exactly one SSL server certificate on the listener. For more information, see TLS listeners for your Network Load Balancer (p. 20).

To support both TCP and UDP on the same port, create a TCP_UDP listener. The target groups for a TCP_UDP listener must use the TCP_UDP protocol.

You can use WebSockets with your listeners.

All network traffic sent to a configured listener is classified as intended traffic. Network traffic that does not match a configured listener is classified as unintended traffic. ICMP requests other than Type 3 are also considered unintended traffic. Network Load Balancers drop unintended traffic without forwarding it to any targets. TCP data packets sent to the listener port for a configured listeners that are not new connections or part of an active TCP connection are rejected with a TCP reset (RST).

## Listener rules

When you create a listener, you specify a rule for routing requests. This rule forwards requests to the specified target group. To update this rule, see Update a listener for your Network Load Balancer (p. 25).

# Create a listener for your Network Load Balancer

A listener is a process that checks for connection requests. You define a listener when you create your load balancer, and you can add listeners to your load balancer at any time.

## Prerequisites

- You must specify a target group for the listener rule. For more information, see Create a target group for your Network Load Balancer (p. 37).
- You must specify an SSL certificate for a TLS listener. The load balancer uses the certificate to terminate the connection and decrypt requests from clients before routing them to targets. For more information, see Server certificates (p. 21).

## Add a listener

You configure a listener with a protocol and a port for connections from clients to the load balancer, and a target group for the default listener rule. For more information, see Listener configuration (p. 19).

**To add a listener using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. Choose **Add listener**.
5. For **Protocol : port**, choose **TCP**, **UDP**, **TCP_UDP**, or **TLS**. Keep the default port or type a different port.
6. [TLS listeners] For **ALPN policy**, choose a policy to enable ALPN or choose **None** to disable ALPN. For more information, see ALPN policies (p. 25).
7. For **Default actions**, choose **Add action**, **Forward to** and then choose an available target group.
8. [TLS listeners] For **Security policy**, we recommend that you keep the default security policy.
9. [TLS listeners] For **Default SSL certificate**, do one of the following:
   - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM** and choose the certificate.
   - If you uploaded a certificate using IAM, choose **From IAM** and choose the certificate.
10. Choose **Save**.
11. [TLS listeners] To add an optional certificate list for use with the SNI protocol, see Add certificates to the certificate list (p. 27).

**To add a listener using the AWS CLI**

Use the create-listener command to create the listener.

# TLS listeners for your Network Load Balancer

To use a TLS listener, you must deploy at least one server certificate on your load balancer. The load balancer uses a server certificate to terminate the front-end connection and then to decrypt requests from clients before sending them to the targets.

Elastic Load Balancing uses a TLS negotiation configuration, known as a security policy, to negotiate TLS connections between a client and the load balancer. A security policy is a combination of protocols and ciphers. The protocol establishes a secure connection between a client and a server and ensures that all data passed between the client and your load balancer is private. A cipher is an encryption algorithm that uses encryption keys to create a coded message. Protocols use several ciphers to encrypt data over the internet. During the connection negotiation process, the client and the load balancer present a list of ciphers and protocols that they each support, in order of preference. The first cipher on the server's list that matches any one of the client's ciphers is selected for the secure connection.

Network Load Balancers do not support TLS renegotiation.

To create a TLS listener, see Add a listener (p. 20). For related demos, see TLS Support on Network Load Balancer and SNI Support on Network Load Balancer.

# Server certificates

The load balancer requires X.509 certificates (server certificate). Certificates are a digital form of identification issued by a certificate authority (CA). A certificate contains identification information, a validity period, a public key, a serial number, and the digital signature of the issuer.

When you create a certificate for use with your load balancer, you must specify a domain name.

We recommend that you create certificates for your load balancers using AWS Certificate Manager (ACM). ACM integrates with Elastic Load Balancing so that you can deploy the certificate on your load balancer. For more information, see the AWS Certificate Manager User Guide.

Alternatively, you can use TLS tools to create a certificate signing request (CSR), then get the CSR signed by a CA to produce a certificate, then import the certificate into ACM or upload the certificate to AWS Identity and Access Management (IAM). For more information, see Importing certificates in the *AWS Certificate Manager User Guide* or Working with server certificates in the *IAM User Guide*.

**Important**
You cannot install certificates with RSA keys larger than 2048-bit or EC keys on your Network Load Balancer.

## Default certificate

When you create a TLS listener, you must specify exactly one certificate. This certificate is known as the *default certificate*. You can replace the default certificate after you create the TLS listener. For more information, see Replace the default certificate (p. 26).

If you specify additional certificates in a certificate list (p. 21), the default certificate is used only if a client connects without using the Server Name Indication (SNI) protocol to specify a hostname or if there are no matching certificates in the certificate list.

If you do not specify additional certificates but need to host multiple secure applications through a single load balancer, you can use a wildcard certificate or add a Subject Alternative Name (SAN) for each additional domain to your certificate.

## Certificate list

After you create a TLS listener, it has a default certificate and an empty certificate list. You can optionally add certificates to the certificate list for the listener. Using a certificate list enables the load balancer to support multiple domains on the same port and provide a different certificate for each domain. For more information, see Add certificates to the certificate list (p. 27).

The load balancer uses a smart certificate selection algorithm with support for SNI. If the hostname provided by a client matches a single certificate in the certificate list, the load balancer selects this

certificate. If a hostname provided by a client matches multiple certificates in the certificate list, the load balancer selects the best certificate that the client can support. Certificate selection is based on the following criteria in the following order:

- Public key algorithm (prefer ECDSA over RSA)
- Hashing algorithm (prefer SHA over MD5)
- Key length (prefer the largest)
- Validity period

The load balancer access log entries indicate the hostname specified by the client and the certificate presented to the client. For more information, see Access log entries (p. 60).

## Certificate renewal

Each certificate comes with a validity period. You must ensure that you renew or replace each certificate for your load balancer before its validity period ends. This includes the default certificate and certificates in a certificate list. Renewing or replacing a certificate does not affect in-flight requests that were received by the load balancer node and are pending routing to a healthy target. After a certificate is renewed, new requests use the renewed certificate. After a certificate is replaced, new requests use the new certificate.

You can manage certificate renewal and replacement as follows:

- Certificates provided by AWS Certificate Manager and deployed on your load balancer can be renewed automatically. ACM attempts to renew certificates before they expire. For more information, see Managed renewal in the *AWS Certificate Manager User Guide*.
- If you imported a certificate into ACM, you must monitor the expiration date of the certificate and renew it before it expires. For more information, see Importing certificates in the *AWS Certificate Manager User Guide*.
- If you imported a certificate into IAM, you must create a new certificate, import the new certificate to ACM or IAM, add the new certificate to your load balancer, and remove the expired certificate from your load balancer.

# Security policies

When you create a TLS listener, you must select a security policy. You can update the security policy as needed. For more information, see Update the security policy (p. 28).

You can choose the security policy that is used for front-end connections. The `ELBSecurityPolicy-2016-08` security policy is always used for backend connections. Network Load Balancers do not support custom security policies.

Elastic Load Balancing provides the following security policies for Network Load Balancers:

- `ELBSecurityPolicy-2016-08` (default)
- `ELBSecurityPolicy-TLS-1-0-2015-04`
- `ELBSecurityPolicy-TLS-1-1-2017-01`
- `ELBSecurityPolicy-TLS-1-2-2017-01`
- `ELBSecurityPolicy-TLS-1-2-Ext-2018-06`
- `ELBSecurityPolicy-FS-2018-06`
- `ELBSecurityPolicy-FS-1-1-2019-08`

- `ELBSecurityPolicy-FS-1-2-2019-08`
- `ELBSecurityPolicy-FS-1-2-Res-2019-08`
- `ELBSecurityPolicy-2015-05` (identical to `ELBSecurityPolicy-2016-08`)
- `ELBSecurityPolicy-FS-1-2-Res-2020-10`

We recommend the `ELBSecurityPolicy-2016-08` policy for compatibility. You can use one of the `ELBSecurityPolicy-FS` policies if you require Forward Secrecy (FS). You can use one of the `ELBSecurityPolicy-TLS` policies to meet compliance and security standards that require disabling certain TLS protocol versions, or to support legacy clients that require deprecated ciphers. Only a small percentage of internet clients require TLS version 1.0. To view the TLS protocol version for requests to your load balancer, enable access logging for your load balancer and examine the access logs. For more information, see Access Logs (p. 58).

## FS security policies

The following table describes the default policy and the `ELBSecurityPolicy-FS` policies. `ELBSecurityPolicy-` has been removed from policy names in the heading row so that they fit.

| | Default | FS-1-2-Res-2020-10 | FS-1-2-Res-2019-08 | FS-1-2-2019-08 | FS-1-1-2019-08 | FS-2018-06 |
|---|---|---|---|---|---|---|
| **TLS Protocols** | | | | | | |
| Protocol-TLSv1 | ✔ | | | | | ✔ |
| Protocol-TLSv1.1 | ✔ | | | | ✔ | ✔ |
| Protocol-TLSv1.2 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| **TLS Ciphers** | | | | | | |
| ECDHE-ECDSA-AES128-GCM-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES128-GCM-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES128-SHA256 | ✔ | | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES128-SHA256 | ✔ | | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES128-SHA | ✔ | | | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES128-SHA | ✔ | | | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES256-GCM-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-GCM-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES256-SHA384 | ✔ | | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-SHA384 | ✔ | | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-SHA | ✔ | | | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES256-SHA | ✔ | | | ✔ | ✔ | ✔ |
| AES128-GCM-SHA256 | ✔ | | | | | |
| AES128-SHA256 | ✔ | | | | | |
| AES128-SHA | ✔ | | | | | |
| AES256-GCM-SHA384 | ✔ | | | | | |
| AES256-SHA256 | ✔ | | | | | |
| AES256-SHA | ✔ | | | | | |

# TLS security policies

The following table describes the default policy and the `ELBSecurityPolicy-TLS` polices. `ELBSecurityPolicy-` has been removed from policy names in the heading row so that they fit.

| | Default | TLS-1-2-Ext-2018-06 | TLS-1-2-2017-01 | TLS-1-1-2017-01 | TLS-1-0-2015-04* |
|---|---|---|---|---|---|
| **TLS Protocols** | | | | | |
| Protocol-TLSv1 | ✔ | | | | ✔ |
| Protocol-TLSv1.1 | ✔ | | | ✔ | ✔ |
| Protocol-TLSv1.2 | ✔ | ✔ | ✔ | ✔ | ✔ |
| **TLS Ciphers** | | | | | |
| ECDHE-ECDSA-AES128-GCM-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES128-GCM-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES128-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES128-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES128-SHA | ✔ | ✔ | | ✔ | ✔ |
| ECDHE-RSA-AES128-SHA | ✔ | ✔ | | ✔ | ✔ |
| ECDHE-ECDSA-AES256-GCM-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-GCM-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-ECDSA-AES256-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ |
| ECDHE-RSA-AES256-SHA | ✔ | ✔ | | ✔ | ✔ |
| ECDHE-ECDSA-AES256-SHA | ✔ | ✔ | | ✔ | ✔ |
| AES128-GCM-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| AES128-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| AES128-SHA | ✔ | ✔ | | ✔ | ✔ |
| AES256-GCM-SHA384 | ✔ | ✔ | ✔ | ✔ | ✔ |
| AES256-SHA256 | ✔ | ✔ | ✔ | ✔ | ✔ |
| AES256-SHA | ✔ | ✔ | | ✔ | ✔ |
| DES-CBC3-SHA | | | | | ✔ |

*Do not use this policy unless you must support a legacy client that requires the DES-CBC3-SHA cipher, which is a weak cipher.

To view the configuration of a security policy for your load balancer using the AWS CLI, use the describe-ssl-policies command.

## ALPN policies

Application-Layer Protocol Negotiation (ALPN) is a TLS extension that is sent on the initial TLS handshake hello messages. ALPN enables the application layer to negotiate which protocols should be used over a secure connection, such as HTTP/1 and HTTP/2.

When the client initiates an ALPN connection, the load balancer compares the client ALPN preference list with its ALPN policy. If the client supports a protocol from the ALPN policy, the load balancer establishes the connection based on the preference list of the ALPN policy. Otherwise, the load balancer does not use ALPN.

### Requirements

- TLS listener
- TLS target group

### Supported ALPN Policies

The following are the supported ALPN policies:

`HTTP1Only`

Negotiate only HTTP/1.*. The ALPN preference list is http/1.1, http/1.0.

`HTTP2Only`

Negotiate only HTTP/2. The ALPN preference list is h2.

`HTTP2Optional`

Prefer HTTP/1.* over HTTP/2 (which can be useful for HTTP/2 testing). The ALPN preference list is http/1.1, http/1.0, h2.

`HTTP2Preferred`

Prefer HTTP/2 over HTTP/1.*. The ALPN preference list is h2, http/1.1, http/1.0.

`None`

Do not negotiate ALPN. This is the default.

### Enable ALPN Connections

You can enable ALPN connections when you create or modify a TLS listener. For more information, see Add a listener (p. 20) and Update the ALPN policy (p. 28).

# Update a listener for your Network Load Balancer

You can update the listener port, listener protocol, or the default listener rule.

The default listener rule forwards requests to the specified target group.

If you change the protocol from TCP or UDP to TLS, you must specify a security policy and server certificate. If you change the protocol from TLS to TCP or UDP, the security policy and server certificate are removed.

**To update your listener using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**.
4. Select the check box for the listener and then choose **Edit**.
5. (Optional) Change the specified values for **Protocol : port**.
6. (Optional) Click the pencil icon to select a different target group for **Default action**.
7. Choose **Update**.

**To update your listener using the AWS CLI**

Use the modify-listener command.

# Update a TLS listener for your Network Load Balancer

After you create a TLS listener, you can replace the default certificate, add or remove certificates from the certificate list, update the security policy, or update the ALPN policy.

**Limitation**

You cannot install certificates with RSA keys larger than 2048-bit or EC keys on your Network Load Balancer.

**Tasks**

- Replace the default certificate (p. 26)
- Add certificates to the certificate list (p. 27)
- Remove certificates from the certificate list (p. 27)
- Update the security policy (p. 28)
- Update the ALPN policy (p. 28)

## Replace the default certificate

You can replace the default certificate for your TLS listener using the following procedure. For more information, see Default certificate (p. 21).

**To replace the default certificate using the 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. Select the load balancer and choose **Listeners**.
4. Select the check box for the listener and choose **Edit**.
5. For **Default SSL certificate**, do one of the following:
   - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM** and choose the certificate.
   - If you uploaded a certificate using IAM, choose **From IAM** and choose the certificate.
6. Choose **Update**.

**To replace the default certificate using the AWS CLI**

Use the modify-listener command with the **--certificates** option.

# Add certificates to the certificate list

You can add certificates to the certificate list for your listener using the following procedure. When you first create a TLS listener, the certificate list is empty. You can add one or more certificates. You can optionally add the default certificate to ensure that this certificate is used with the SNI protocol even if it is replaced as the default certificate. For more information, see Certificate list (p. 21).

**To add certificates to the certificate list using the 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. Select the load balancer and choose **Listeners**.
4. For the HTTPS listener to update, choose **View/edit certificates**, which displays the default certificate followed by any other certificates that you've added to the listener.
5. Choose the **Add certificates** icon (the plus sign) in the menu bar, which displays the default certificate followed by any other certificates managed by ACM and IAM. If you've already added a certificate to the listener, its check box is selected and disabled.
6. To add certificates that are already managed by ACM or IAM, select the check boxes for the certificates and choose **Add**.
7. If you have a certificate that isn't managed by ACM or IAM, import it to ACM and add it to your listener as follows:
   a. Choose **Import certificate**.
   b. For **Certificate private key**, paste the PEM-encoded, unencrypted private key for the certificate.
   c. For **Certificate body**, paste the PEM-encoded certificate.
   d. (Optional) For **Certificate chain**, paste the PEM-encoded certificate chain.
   e. Choose **Import**. The newly imported certificate appears in the list of available certificates and is selected.
   f. Choose **Add**.
8. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

**To add a certificate to the certificate list using the AWS CLI**

Use the add-listener-certificates command.

# Remove certificates from the certificate list

You can remove certificates from the certificate list for a TLS listener using the following procedure. To remove the default certificate for a TLS listener, see Replace the default certificate (p. 26).

**To remove certificates from the certificate list using the 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. Select the load balancer and choose **Listeners**.
4. For the listener to update, choose **View/edit certificates**, which displays the default certificate followed by any other certificates that you've added to the listener.
5. Choose the **Remove certificates** icon (the minus sign) in the menu bar.

6. Select the check boxes for the certificates and choose **Remove**.
7. To leave this screen, choose the **Back to the load balancer** icon (the back button) in the menu bar.

**To remove a certificate from the certificate list using the AWS CLI**

Use the remove-listener-certificates command.

## Update the security policy

When you create a TLS listener, you can select the security policy that meets your needs. When a new security policy is added, you can update your TLS listener to use the new security policy. Network Load Balancers do not support custom security policies. For more information, see Security policies (p. 22).

**To update the security policy using the 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. Select the load balancer and choose **Listeners**.
4. Select the check box for the TLS listener and choose **Edit**.
5. For **Security policy**, choose a security policy.
6. Choose **Update**.

**To update the security policy using the AWS CLI**

Use the modify-listener command with the **--ssl-policy** option.

## Update the ALPN policy

You can update the ALPN policy for your TLS listener using the following procedure. For more information, see ALPN policies (p. 25).

**To update the ALPN policy using the 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. Select the load balancer and choose **Listeners**.
4. Select the check box for the TLS listener and choose **Edit**.
5. For **ALPN policy**, choose a policy to enable ALPN or choose **None** to disable ALPN.
6. Choose **Update**.

**To update the ALPN policy using the AWS CLI**

Use the modify-listener command with the **--alpn-policy** option.

# Delete a listener for your Network Load Balancer

You can delete a listener at any time.

**To delete a listener using the console**

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

2. In the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select the load balancer and choose **Listeners**. Select the check box for the listener, and then choose **Delete**.
4. When prompted for confirmation, choose **Yes, Delete**.

**To delete a listener using the AWS CLI**

Use the delete-listener command.

# Target groups for your Network Load Balancers

Each *target group* is used to route requests to one or more registered targets. When you create a listener, you specify a target group for its default action. Traffic is forwarded to the target group specified in the listener rule. You can create different target groups for different types of requests. For example, create one target group for general requests and other target groups for requests to the microservices for your application. For more information, see Network Load Balancer components (p. 1).

You define health check settings for your load balancer on a per target group basis. Each target group uses the default health check settings, unless you override them when you create the target group or modify them later on. After you specify a target group in a rule for a listener, the load balancer continually monitors the health of all targets registered with the target group that are in an Availability Zone enabled for the load balancer. The load balancer routes requests to the registered targets that are healthy. For more information, see Health checks for your target groups (p. 38).

**Contents**



## Routing configuration

By default, a load balancer routes requests to its targets using the protocol and port number that you specified when you created the target group. Alternatively, you can override the port used for routing traffic to a target when you register it with the target group.

Target groups for Network Load Balancers support the following protocols and ports:

- **Protocols**: TCP, TLS, UDP, TCP_UDP
- **Ports**: 1-65535

If a target group is configured with the TLS protocol, the load balancer establishes TLS connections with the targets using certificates that you install on the targets. The load balancer does not validate these certificates. Therefore, you can use self-signed certificates or certificates that have expired. Because the load balancer is in a virtual private cloud (VPC), traffic between the load balancer and the targets is authenticated at the packet level, so it is not at risk of man-in-the-middle attacks or spoofing even if the certificates on the targets are not valid.

The following table summarizes the supported combinations of listener protocol and target group settings.

| Listener protocol | Target group protocol | Target group type | Health check protocol |
|---|---|---|---|
| TCP | TCP \| TCP_UDP | instance \| ip | HTTP \| HTTPS \| TCP |
| TLS | TCP \| TLS | instance \| ip | HTTP \| HTTPS \| TCP |
| UDP | UDP \| TCP_UDP | instance \| ip | HTTP \| HTTPS \| TCP |
| TCP_UDP | TCP_UDP | instance \| ip | HTTP \| HTTPS \| TCP |

# Target type

When you create a target group, you specify its target type, which determines how you specify its targets. After you create a target group, you cannot change its target type.

The following are the possible target types:

`instance`

The targets are specified by instance ID.

`ip`

The targets are specified by IP address.

When the target type is `ip`, you can specify IP addresses from one of the following CIDR blocks:

- The subnets of the VPC for the target group
- 10.0.0.0/8 (RFC 1918)
- 100.64.0.0/10 (RFC 6598)
- 172.16.0.0/12 (RFC 1918)
- 192.168.0.0/16 (RFC 1918)

**Important**
You can't specify publicly routable IP addresses.

These supported CIDR blocks enable you to register the following with a target group: ClassicLink instances, AWS resources that are addressable by IP address and port (for example, databases), and on-premises resources linked to AWS through AWS Direct Connect or a Site-to-Site VPN connection.

When the target type is `ip`, the load balancer can support 55,000 simultaneous connections or about 55,000 connections per minute to each unique target (IP address and port). If you exceed these connections, there is an increased chance of port allocation errors. If you get port allocation errors, add more targets to the target group.

Network Load Balancers do not support the `lambda` target type, only Application Load Balancers support the `lambda` target type. For more information, see Lambda functions as targets in the *User Guide for Application Load Balancers*.

If you have micro services on instances registered with a Network Load Balancer, you cannot use the load balancer to provide communication between them unless the load balancer is internet-facing or the instances are registered by IP address. For more information, see Connections time out for requests from a target to its load balancer (p. 68).

## Request routing and IP addresses

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. The load balancer rewrites the destination IP address from the data packet before forwarding it to the target instance.

If you specify targets using IP addresses, you can route traffic to an instance using any private IP address from one or more network interfaces. This enables multiple applications on an instance to use the same port. Note that each network interface can have its own security group. The load balancer rewrites the destination IP address before forwarding it to the target.

For more information allowing traffic to your instances, see Target security groups (p. 43).

## Source IP preservation

If you specify targets by instance ID, the source IP addresses of the clients are preserved and provided to your applications.

If you specify targets by IP address, the source IP addresses provided depend on the protocol of the target group as follows:

- TCP and TLS: The source IP addresses are the private IP addresses of the load balancer nodes. If you need the IP addresses of the clients, enable proxy protocol (p. 34) on the load balancer and get the client IP addresses from the proxy protocol header.
- UDP and TCP_UDP: The source IP addresses are the IP addresses of the clients.

If you are using a Network Load Balancer with a VPC endpoint service or with AWS Global Accelerator, the source IP addresses provided to your application are the private IP addresses of the load balancer nodes. If you need the IP addresses of the service consumers, enable proxy protocol (p. 34) on the load balancer.

If you specify targets by instance ID, you might encounter TCP/IP connection limitations related to observed socket reuse on the targets. These connection limitations can occur when a client, or a NAT device in front of the client, uses the same source IP address and source port when connecting to multiple load balancer nodes simultaneously. If the load balancer routes the connections to the same target, these connections appear to the target as if they come from the same source socket, which results in connection errors. If this happens, the clients can retry if the connection fails or reconnect if the connection is interrupted. You can reduce this type of connection error by increasing the number of source ephemeral ports or by increasing the number of targets for the load balancer. You can prevent this type of connection error by specifying targets by IP address or by disabling cross-zone load balancing.

# Registered targets

Your load balancer serves as a single point of contact for clients and distributes incoming traffic across its healthy registered targets. Each target group must have at least one registered target in each Availability Zone that is enabled for the load balancer. You can register each target with one or more target groups.

If demand on your application increases, you can register additional targets with one or more target groups in order to handle the demand. The load balancer starts routing traffic to a newly registered target as soon as the registration process completes.

If demand on your application decreases, or you need to service your targets, you can deregister targets from your target groups. Deregistering a target removes it from your target group, but does not affect the target otherwise. The load balancer stops routing traffic to a target as soon as it is deregistered. The

target enters the `draining` state until in-flight requests have completed. You can register the target with the target group again when you are ready for it to resume receiving traffic.

If you are registering targets by instance ID, you can use your load balancer with an Auto Scaling group. After you attach a target group to an Auto Scaling group, Auto Scaling registers your targets with the target group for you when it launches them. For more information, see Attaching a load balancer to your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.

**Requirements**

- You cannot register instances by instance ID if they use one of the following instance types: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, or T1.
- You cannot register instances by instance ID if they are in a VPC that is peered to the load balancer VPC (same Region or different Region). You can register these instances by IP address.
- If you register a target by IP address and the IP address is in the same VPC as the load balancer, the load balancer verifies that it is from a subnet that it can reach.
- For UDP and TCP_UDP target groups, do not register instances by IP address if they reside outside of the load balancer VPC or if they use one of the following instance types: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, or T1. Targets that reside outside the load balancer VPC or use an unsupported instance type might be able to receive traffic from the load balancer but then be unable to respond.

# Target group attributes

The following are the target group attributes:

`deregistration_delay.timeout_seconds`

The amount of time for Elastic Load Balancing to wait before changing the state of a deregistering target from `draining` to `unused`. The range is 0-3600 seconds. The default value is 300 seconds.

`deregistration_delay.connection_termination.enabled`

Indicates whether the load balancer terminates connections at the end of the deregistration timeout. The value is `true` or `false`. The default is `false`.

`proxy_protocol_v2.enabled`

Indicates whether proxy protocol version 2 is enabled. By default, proxy protocol is disabled.

`stickiness.enabled`

Indicates whether sticky sessions are enabled.

`stickiness.type`

The type of stickiness. The possible value is `source_ip`.

# Deregistration delay

When you deregister a target, the load balancer stops creating new connections to the target. The load balancer uses connection draining to ensure that in-flight traffic completes on the existing connections. If the deregistered target stays healthy and an existing connection is not idle, the load balancer can continue to send traffic to the target. To ensure that existing connections are closed, you can do one of the following: enable the target group attribute for connection termination, ensure that the instance is unhealthy before you deregister it, or periodically close client connections.

The initial state of a deregistering target is `draining`. By default, the load balancer changes the state of a deregistering target to `unused` after 300 seconds. To change the amount of time that the load

balancer waits before changing the state of a deregistering target to `unused`, update the deregistration delay value. We recommend that you specify a value of at least 120 seconds to ensure that requests are completed.

If you enable the target group attribute for connection termination, connections to deregistered targets are closed shortly after the end of the deregistration timeout.

New console

**To update the deregistration attributes using the new console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** page, in the **Attributes** section, choose **Edit**.
5. To change the deregistration timeout, enter a new value for **Deregistration delay**. To ensure that existing connections are closed after you deregister targets, select **Connection termination on deregistration**.
6. Choose **Save changes**.

Old console

**To update the deregistration attributes using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group and choose **Description**, **Edit attributes**.
4. To change the deregistration timeout, enter a new value for **Deregistration delay**. To ensure that existing connections are closed after you deregister targets, select **Connection termination on deregistration**.
5. Choose **Save**.

**To update the deregistration attributes using the AWS CLI**

Use the modify-target-group-attributes command.

# Proxy protocol

Network Load Balancers use proxy protocol version 2 to send additional connection information such as the source and destination. Proxy protocol version 2 provides a binary encoding of the proxy protocol header. The load balancer prepends a proxy protocol header to the TCP data. It does not discard or overwrite any existing data, including any proxy protocol headers sent by the client or any other proxies, load balancers, or servers in the network path. Therefore, it is possible to receive more than one proxy protocol header. Also, if there is another network path to your targets outside of your Network Load Balancer, the first proxy protocol header might not be the one from your Network Load Balancer.

If you specify targets by IP address, the source IP addresses provided to your applications depend on the protocol of the target group as follows:

- TCP and TLS: The source IP addresses are the private IP addresses of the load balancer nodes. If you need the IP addresses of the clients, enable proxy protocol and get the client IP addresses from the proxy protocol header.
- UDP and TCP_UDP: The source IP addresses are the IP addresses of the clients.

If you specify targets by instance ID, the source IP addresses provided to your applications are the client IP addresses. However, if you prefer, you can enable proxy protocol and get the client IP addresses from the proxy protocol header.

# Health check connections

After you enable proxy protocol, the proxy protocol header is also included in health check connections from the load balancer. However, with health check connections, the client connection information is not sent in the proxy protocol header.

# VPC endpoint services

For traffic coming from service consumers through a VPC endpoint service, the source IP addresses provided to your applications are the private IP addresses of the load balancer nodes. If your applications need the IP addresses of the service consumers, enable proxy protocol and get them from the proxy protocol header.

The proxy protocol header also includes the ID of the endpoint. This information is encoded using a custom Type-Length-Value (TLV) vector as follows.

| Field | Length (in octets) | Description |
|---|---|---|
| Type | 1 | PP2_TYPE_AWS (0xEA) |
| Length | 2 | The length of value |
| Value | 1 | PP2_SUBTYPE_AWS_VPCE_ID (0x01) |
| | variable (value length minus 1) | The ID of the endpoint |

For an example that parses TLV type 0xEA, see https://github.com/aws/elastic-load-balancing-tools/tree/master/proprot.

# Enable proxy protocol

Before you enable proxy protocol on a target group, make sure that your applications expect and can parse the proxy protocol v2 header, otherwise, they might fail. For more information, see PROXY protocol versions 1 and 2.

New console

**To enable proxy protocol v2 using the new console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name the target group to open its details page.
4. On the **Group details** page, in the **Attributes** section, choose **Edit**.
5. On the **Edit attributes** page, select **Proxy protocol v2**.
6. Choose **Save changes**.

Old console

**To enable proxy protocol v2 using the old console**

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

2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. Choose **Description**, **Edit attributes**.
5. For **Proxy protocol v2**, choose **Enable**.
6. Choose **Save**.

**To enable proxy protocol v2 using the AWS CLI**

Use the modify-target-group-attributes command.

# Sticky sessions

Sticky sessions are a mechanism to route client traffic to the same target in a target group. This is useful for servers that maintain state information in order to provide a continuous experience to clients.

### Considerations

- Using sticky sessions can lead to an uneven distribution of connections and flows, which might impact the availability of your targets. For example, all clients behind the same NAT device have the same source IP address. Therefore, all traffic from these clients is routed to the same target.
- The load balancer might reset the sticky sessions for a target group if the health state of any of its targets changes or if you register or deregister targets with the target group.
- Sticky sessions are not supported with TLS listeners and TLS target groups.

New console

**To enable sticky sessions using the new console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** page, in the **Attributes** section, choose **Edit**.
5. On the **Edit attributes** page, select **Stickiness**.
6. Choose **Save changes**.

Old console

**To enable sticky sessions using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. Choose **Description**, **Edit attributes**.
5. For **Stickiness**, select **Enable**.
6. Choose **Save**.

**To enable sticky sessions using the AWS CLI**

Use the modify-target-group-attributes command with the `stickiness.enabled` attribute.

# Create a target group for your Network Load Balancer

You register targets for your Network Load Balancer with a target group. By default, the load balancer sends requests to registered targets using the port and protocol that you specified for the target group. You can override this port when you register each target with the target group.

After you create a target group, you can add tags.

To route traffic to the targets in a target group, create a listener and specify the target group in the default action for the listener. For more information, see Listener rules (p. 19).

You can add or remove targets from your target group at any time. For more information, see Register targets with your target group (p. 43). You can also modify the health check settings for your target group. For more information, see Modify the health check settings of a target group (p. 42).

New console

### To create a target group using the new console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Choose a target type**, select **Instances** to register targets by instance ID or **IP addresses** to register targets by IP address.
5. For **Target group name**, type a name for the target group. This name must be unique per region per account, can have a maximum of 32 characters, must contain only alphanumeric characters or hyphens, and must not begin or end with a hyphen.
6. For **Protocol**, choose a protocol as follows:
   - If the listener protocol is TCP, choose **TCP** or **TCP_UDP**.
   - If the listener protocol is TLS, choose **TCP** or **TLS**.
   - If the listener protocol is UDP, choose **UDP** or **TCP_UDP**.
   - If the listener protocol is TCP_UDP, choose **TCP_UDP**.
7. (Optional) For **Port**, modify the default value as needed.
8. For **VPC**, select a virtual private cloud (VPC).
9. (Optional) In the **Health checks** section, modify the default settings as needed.
10. (Optional) Add one or more tags as follows:
    a. Expand the **Tags** section.
    b. Choose **Add tag**.
    c. Enter the tag key and tag value.
11. Choose **Next**.
12. (Optional) Add one or more targets as follows:
    - If the target type is **Instances**, select one or more instances, enter one or more ports, and then choose **Include as pending below**.
    - If the target type is **IP addresses**, select the network, enter the IP address and ports, and then choose **Include as pending below**.
13. Choose **Create target group**.
14. (Optional) You can specify the target group in the default listener rule. For more information, see Create a Listener (p. 20) and Update a Listener (p. 25).

Old console

**To create a target group using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Target group name**, type a name for the target group. This name must be unique per region per account, can have a maximum of 32 characters, must contain only alphanumeric characters or hyphens, and must not begin or end with a hyphen.
5. For **Protocol**, choose a protocol as follows:
   - If the listener protocol is TCP, choose **TCP** or **TCP_UDP**.
   - If the listener protocol is TLS, choose **TCP** or **TLS**.
   - If the listener protocol is UDP, choose **UDP** or **TCP_UDP**.
   - If the listener protocol is TCP_UDP, choose **TCP_UDP**.
6. (Optional) For **Port**, modify the default value as needed.
7. For **Target type**, select `instance` to specify targets by instance ID or `ip` to specify targets by IP address.
8. For **VPC**, select a virtual private cloud (VPC).
9. (Optional) For **Health check settings** and **Advanced health check settings**, modify the default settings as needed. Choose **Create**.
10. (Optional) Add one or more tags as follows:
    a. Select the newly created target group.
    b. Choose **Tags**, **Add/Edit Tags**.
    c. On the **Add/Edit Tags** page, for each tag that you add, choose **Create Tag** and then specify the tag key and tag value. When you have finished adding tags, choose **Save**.
11. (Optional) To add targets to the target group, see Register targets with your target group (p. 43).
12. (Optional) You can specify the target group in the default listener rule. For more information, see Create a Listener (p. 20) and Update a Listener (p. 25).

**To create a target group using the AWS CLI**

Use the create-target-group command to create the target group, the add-tags command to tag your target group, and the register-targets command to add targets.

# Health checks for your target groups

You register your targets with one or more target groups. The load balancer starts routing requests to a newly registered target as soon as the registration process completes. It can take a few minutes for the registration process to complete and health checks to start.

Network Load Balancers use active and passive health checks to determine whether a target is available to handle requests. By default, each load balancer node routes requests only to the healthy targets in its Availability Zone. If you enable cross-zone load balancing, each load balancer node routes requests to the healthy targets in all enabled Availability Zones. For more information, see Cross-zone load balancing (p. 11).

With active health checks, the load balancer periodically sends a request to each registered target to check its status. Each load balancer node checks the health of each target, using the health check

settings for the target group with which the target is registered. After each health check is completed, the load balancer node closes the connection that was established for the health check.

With passive health checks, the load balancer observes how targets respond to connections. Passive health checks enable the load balancer to detect an unhealthy target before it is reported as unhealthy by the active health checks. You cannot disable, configure, or monitor passive health checks. Passive health checks are not supported for UDP traffic.

If a target becomes unhealthy, the load balancer sends a TCP RST for packets received on the client connections associated with the target.

If target groups don't have a healthy target in an enabled Availability Zone, we remove the IP address for the corresponding subnet from DNS so that requests cannot be routed to targets in that Availability Zone. If all targets fail health checks at the same time in all enabled Availability Zones, the load balancer fails open. The effect of the fail open is to allow traffic to all targets in all enabled Availability Zones, regardless of their health status.

For HTTP or HTTPS health check requests, the host header contains the IP address of the load balancer node and the listener port, not the IP address of the target and the health check port.

If you add a TLS listener to your Network Load Balancer, we perform a listener connectivity test. As TLS termination also terminates a TCP connection, a new TCP connection is established between your load balancer and your targets. Therefore, you might see the TCP pings for this test sent from your load balancer to the targets that are registered with your TLS listener. You can identify these TCP pings because they have the source IP address of your Network Load Balancer and the connections do not contain data packets.

For a UDP service, target availability can be tested using non-UDP health checks on your target group. You can use any available health check (TCP, HTTP, or HTTPS), and any port on your target to verify the availability of a UDP service. If the service receiving the health check fails, your target is considered unavailable. To improve the accuracy of health checks for a UDP service, configure the service listening to the health check port to track the status of your UDP service and fail the health check if the service is unavailable.

# Health check settings

You configure active health checks for the targets in a target group using the following settings. If the health checks exceed **UnhealthyThresholdCount** consecutive failures, the load balancer takes the target out of service. When the health checks exceed **HealthyThresholdCount** consecutive successes, the load balancer puts the target back in service.

| Setting | Description |
| --- | --- |
| **HealthCheckProtocol** | The protocol the load balancer uses when performing health checks on targets. The possible protocols are HTTP, HTTPS, and TCP. The default is the TCP protocol. |
| **HealthCheckPort** | The port the load balancer uses when performing health checks on targets. The default is to use the port on which each target receives traffic from the load balancer. |
| **HealthCheckPath** | [HTTP/HTTPS health checks] The ping path that is the destination on the targets for health checks. The default is /. |
| **HealthCheckTimeoutSeconds** | The amount of time, in seconds, during which no response from a target means a failed health |

| Setting | Description |
| --- | --- |
| | check. This value must be 6 seconds for HTTP health checks and 10 seconds for TCP and HTTPS health checks. |
| **HealthCheckIntervalSeconds** | The approximate amount of time, in seconds, between health checks of an individual target. This value can be 10 seconds or 30 seconds. The default is 30 seconds.<br>**Important**<br>Health checks for a Network Load Balancer are distributed and use a consensus mechanism to determine target health. Therefore, targets receive more than the configured number of health checks. To reduce the impact to your targets if you are using HTTP health checks, use a simpler destination on the targets, such as a static HTML file, or switch to TCP health checks. |
| **HealthyThresholdCount** | The number of consecutive successful health checks required before considering an unhealthy target healthy. The range is 2 to 10. The default is 3. |
| **UnhealthyThresholdCount** | The number of consecutive failed health checks required before considering a target unhealthy. This value must be the same as the healthy threshold count. |
| **Matcher** | [HTTP/HTTPS health checks] The HTTP codes to use when checking for a successful response from a target. This value must be 200 to 399. |

# Target health status

Before the load balancer sends a health check request to a target, you must register it with a target group, specify its target group in a listener rule, and ensure that the Availability Zone of the target is enabled for the load balancer.

The following table describes the possible values for the health status of a registered target.

| Value | Description |
| --- | --- |
| `initial` | The load balancer is in the process of registering the target or performing the initial health checks on the target.<br><br>Related reason codes: `Elb.RegistrationInProgress` \| `Elb.InitialHealthChecking` |
| `healthy` | The target is healthy.<br><br>Related reason codes: None |

| Value | Description |
| --- | --- |
| `unhealthy` | The target did not respond to a health check or failed the health check.<br><br>Related reason code: `Target.FailedHealthChecks` |
| `unused` | The target is not registered with a target group, the target group is not used in a listener rule, the target is in an Availability Zone that is not enabled, or the target is in the stopped or terminated state.<br><br>Related reason codes: `Target.NotRegistered` \| `Target.NotInUse` \| `Target.InvalidState` \| `Target.IpUnusable` |
| `draining` | The target is deregistering and connection draining is in process.<br><br>Related reason code: `Target.DeregistrationInProgress` |
| `unavailable` | Target health is unavailable.<br><br>Related reason code: `Elb.InternalError` |

# Health check reason codes

If the status of a target is any value other than `Healthy`, the API returns a reason code and a description of the issue, and the console displays the same description in a tooltip. Note that reason codes that begin with `Elb` originate on the load balancer side and reason codes that begin with `Target` originate on the target side.

| Reason code | Description |
| --- | --- |
| `Elb.InitialHealthChecking` | Initial health checks in progress |
| `Elb.InternalError` | Health checks failed due to an internal error |
| `Elb.RegistrationInProgress` | Target registration is in progress |
| `Target.DeregistrationInProgress` | Target deregistration is in progress |
| `Target.FailedHealthChecks` | Health checks failed |
| `Target.InvalidState` | Target is in the stopped state<br><br>Target is in the terminated state<br><br>Target is in the terminated or stopped state<br><br>Target is in an invalid state |
| `Target.IpUnusable` | The IP address cannot be used as a target, as it is in use by a load balancer |
| `Target.NotInUse` | Target group is not configured to receive traffic from the load balancer |

| Reason code | Description |
| --- | --- |
| | Target is in an Availability Zone that is not enabled for the load balancer |
| `Target.NotRegistered` | Target is not registered to the target group |

# Check the health of your targets

You can check the health status of the targets registered with your target groups.

New console

### To check the health of your targets using the new console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Targets** tab, the **Status** column indicates the status of each target.
5. If the target status is any value other than `Healthy`, the **Status details** column contains more information.

Old console

### To check the health of your targets using the old console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. Choose **Targets**, and view the status of each target in the **Status** column. If the status is any value other than `Healthy`, view the tooltip for more information.

#### To check the health of your targets using the AWS CLI

Use the describe-target-health command. The output of this command contains the target health state. It includes a reason code if the status is any value other than `Healthy`.

#### To receive email notifications about unhealthy targets

Use CloudWatch alarms to trigger a Lambda function to send details about unhealthy targets. For step-by-step instructions, see the following blog post: Identifying unhealthy targets of your load balancer.

# Modify the health check settings of a target group

You can modify some of the health check settings for your target group. If the protocol of the target group is TCP, TLS, UDP, or TCP_UDP, you can't modify the health check protocol, interval, timeout, or success codes.

New console

### To modify health check settings for a target group using the new console

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

2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Group details** tab, in the **Health check settings** section, choose **Edit**.
5. On the **Edit health check settings** page, modify the settings as needed, and then choose **Save changes**.

Old console

**To modify health check settings for a target group using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. Choose **Health checks**, **Edit**.
5. On the **Edit target group** page, modify the settings as needed, and then choose **Save**.

**To modify health check settings for a target group using the AWS CLI**

Use the modify-target-group command.

# Register targets with your target group

When your target is ready to handle requests, you register it with one or more target groups. You can register targets by instance ID or by IP address. The load balancer starts routing requests to the target as soon as the registration process completes and the target passes the initial health checks. It can take a few minutes for the registration process to complete and health checks to start. For more information, see Health checks for your target groups (p. 38).

If demand on your currently registered targets increases, you can register additional targets in order to handle the demand. If demand on your registered targets decreases, you can deregister targets from your target group. It can take a few minutes for the deregistration process to complete and for the load balancer to stop routing requests to the target. If demand increases subsequently, you can register targets that you deregistered with the target group again. If you need to service a target, you can deregister it and then register it again when servicing is complete.

When you deregister a target, Elastic Load Balancing waits until in-flight requests have completed. This is known as *connection draining*. The status of a target is `draining` while connection draining is in progress. After deregistration is complete, status of the target changes to `unused`. For more information, see Deregistration delay (p. 33).

If you are registering targets by instance ID, you can use your load balancer with an Auto Scaling group. After you attach a target group to an Auto Scaling group and the group scales out, the instances launched by the Auto Scaling group are automatically registered with the target group. If you detach the load balancer from the Auto Scaling group, the instances are automatically deregistered from the target group. For more information, see Attaching a Load Balancer to Your Auto Scaling Group in the *Amazon EC2 Auto Scaling User Guide*.

## Target security groups

When you register EC2 instances as targets, you must ensure that the security groups for these instances allow traffic on both the listener port and the health check port.

### Limits

- Network Load Balancers do not have associated security groups. Therefore, the security groups for your targets must use IP addresses to allow traffic from the load balancer.
- You cannot use the security groups for clients as a source in the security groups for the targets. Instead, use the client CIDR blocks as sources in the target security groups.

The following are the recommended rules for instance security groups.

### Recommended rules for instance security groups

| Inbound | | | |
|---|---|---|---|
| **Source** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *target* | *target* | Allow client traffic (`instance` target type) |
| *VPC CIDR* | *target* | *target* | Allow client traffic (`ip` target type) |
| *VPC CIDR* | *health check* | *health check* | Allow health check traffic from the load balancer |

If you register targets by IP address and do not want to grant access to the entire VPC CIDR, you can grant access to the private IP addresses used by the load balancer nodes. There is one IP address per load balancer subnet. To find these addresses, use the following procedure.

### To find the private IP addresses to allow

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Network Interfaces**.
3. In the search field, type the name of your Network Load Balancer. There is one network interface per load balancer subnet.
4. On the **Details** tab for each network interface, copy the address from **Primary private IPv4 IP**.

# Network ACLs

When you register EC2 instances as targets, you must ensure that the network ACLs for the subnets for your instances allow traffic on both the listener port and the health check port. The default network access control list (ACL) for a VPC allows all inbound and outbound traffic. If you create custom network ACLs, verify that they allow the appropriate traffic.

The network ACLs associated with the subnets for your instances must allow the following traffic for an internet-facing load balancer.

### Recommended rules for instance subnets

| Inbound | | | |
|---|---|---|---|
| **Source** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *listener* | *listener* | Allow client traffic (`instance` target type) |

| | | | |
|---|---|---|---|
| *VPC CIDR* | *listener* | *listener* | Allow client traffic (`ip` target type) |
| *VPC CIDR* | *health check* | *health check* | Allow health check traffic from the load balancer |
| **Outbound** | | | |
| **Destination** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *listener* | *listener* | Allow responses to clients (`instance` target type) |
| *VPC CIDR* | *listener* | *listener* | Allow responses to clients (`ip` target type) |
| *VPC CIDR* | *health check* | 1024-65535 | Allow health check traffic |

The network ACLs associated with the subnets for your load balancer must allow the following traffic for an internet-facing load balancer.

**Recommended rules for load balancer subnets**

| | | | |
|---|---|---|---|
| **Inbound** | | | |
| **Source** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *listener* | *listener* | Allow client traffic (`instance` target type ) |
| *VPC CIDR* | *listener* | *listener* | Allow client traffic (`ip` target type ) |
| *VPC CIDR* | *health check* | 1024-65535 | Allow health check traffic |
| **Outbound** | | | |
| **Destination** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *listener* | *listener* | Allow responses to clients (`instance` target type ) |
| *VPC CIDR* | *listener* | *listener* | Allow responses to clients (`ip` target type ) |
| *VPC CIDR* | *health check* | *health check* | Allow health check traffic |
| *VPC CIDR* | *health check* | 1024-65535 | Allow health check traffic |

For an internal load balancer, the network ACLs for the subnets for your instances and load balancer nodes must allow both inbound and outbound traffic to and from the VPC CIDR, on the listener port and ephemeral ports.

# Register or deregister targets

Each target group must have at least one registered target in each Availability Zone that is enabled for the load balancer.

The target type of your target group determines how you register targets with that target group. For more information, see Target type (p. 31).

### Requirements

- You cannot register instances by instance ID if they use one of the following instance types: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, or T1.
- You cannot register instances by instance ID if they are in a VPC that is peered to the load balancer VPC (same Region or different Region). You can register these instances by IP address.
- If you register a target by IP address and the IP address is in the same VPC as the load balancer, the load balancer verifies that it is from a subnet that it can reach.
- For UDP and TCP_UDP target groups, do not register instances by IP address if they reside outside of the load balancer VPC or if they use one of the following instance types: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, or T1. Targets that reside outside the load balancer VPC or use an unsupported instance type might be able to receive traffic from the load balancer but then be unable to respond.

### Contents



## Register or deregister targets by instance ID

An instance must be in the `running` state when you register it.

New console

#### To register or deregister targets by instance ID using the new console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. To register instances, choose **Register targets**. Select one or more instances, enter the default instance port as needed, and then choose **Include as pending below**. When you are finished adding instances, choose **Register pending targets**.
6. To deregister instances, select the instance and then choose **Deregister**.

Old console

#### To register or deregister targets by instance ID using the old console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.

4. Choose **Targets**, **Edit**.
5. (Optional) For **Registered instances**, select any instances to be deregistered and choose **Remove**.
6. (Optional) For **Instances**, select any running instances to be registered, modify the default instance port as needed, and then choose **Add to registered**.
7. Choose **Save**.

## Register or deregister targets by IP address

An IP address that you register must be from one of the following CIDR blocks:

- The subnets of the VPC for the target group
- 10.0.0.0/8 (RFC 1918)
- 100.64.0.0/10 (RFC 6598)
- 172.16.0.0/12 (RFC 1918)
- 192.168.0.0/16 (RFC 1918)

New console

**To register or deregister targets by IP address using the new console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Chose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. To register IP addresses, choose **Register targets**. For each IP address, select the network, Availability Zone, IP address, and port, and then choose **Include as pending below**. When you are finished specifying addresses, choose **Register pending targets**.
6. To deregister IP addresses, select the IP addresses and then choose **Deregister**. If you have many registered IP addresses, you might find it helpful to add a filter or change the sort order.

Old console

**To register or deregister targets by IP address using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group and choose **Targets**, **Edit**.
4. To register IP addresses, choose the **Register targets** icon (the plus sign) in the menu bar. For each IP address, specify the network, Availability Zone, IP address, and port, and then choose **Add to list**. When you are finished specifying addresses, choose **Register**.
5. To deregister IP addresses, choose the **Deregister targets** icon (the minus sign) in the menu bar. If you have many registered IP addresses, you might find it helpful to add a filter or change the sort order. Select the IP addresses and choose **Deregister**.
6. To leave this screen, choose the **Back to target group** icon (the back button) in the menu bar.

## Register or deregister targets using the AWS CLI

Use the register-targets command to add targets and the deregister-targets command to remove targets.

# Tags for your target group

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

You can add multiple tags to each target group. Tag keys must be unique for each target group. If you add a tag with a key that is already associated with the target group, it updates the value of that tag.

When you are finished with a tag, you can remove it.

### Restrictions

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- 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. Tags with this prefix do not count against your tags per resource limit.

New console

#### To update the tags for a target group using the new console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Choose the name of the target group to open its details page.
4. On the **Tags** tab, choose **Manage tags** and do one or more of the following:
   a. To update a tag, enter new values for **Key** and **Value**.
   b. To add a tag, choose **Add tag** and enter values for **Key** and **Value**.
   c. To delete a tag, choose **Remove** next to the tag.
5. When you have finished updating tags, choose **Save changes**.

Old console

#### To update the tags for a target group using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group.
4. On the **Tags** tab, choose **Add/Edit Tags**, and then do one or more of the following:
   a. To update a tag, edit the values of **Key** and **Value**.
   b. To add a new tag, choose **Create Tag** and then type values for **Key** and **Value**.
   c. To delete a tag, choose the delete icon (X) next to the tag.
5. When you have finished updating tags, choose **Save**.

#### To update the tags for a target group using the AWS CLI

Use the add-tags and remove-tags commands.

# Delete a target group

You can delete a target group if it is not referenced by the forward actions of any listener rules. Deleting a target group does not affect the targets registered with the target group. If you no longer need a registered EC2 instance, you can stop or terminate it.

New console

**To delete a target group using the new console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group and choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes, delete**.

Old console

**To delete a target group using the old console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **LOAD BALANCING**, choose **Target Groups**.
3. Select the target group and choose **Actions**, **Delete**.
4. When prompted for confirmation, choose **Yes**.

**To delete a target group using the AWS CLI**

Use the delete-target-group command.

# Monitor your Network Load Balancers

You can use the following features to monitor your load balancers, analyze traffic patterns, and troubleshoot issues with your load balancers and targets.

**CloudWatch metrics**

You can use Amazon CloudWatch to retrieve statistics about data points for your load balancers and targets as an ordered set of time-series data, known as *metrics*. You can use these metrics to verify that your system is performing as expected. For more information, see CloudWatch metrics for your Network Load Balancer (p. 50).

**VPC Flow Logs**

You can use VPC Flow Logs to capture detailed information about the traffic going to and from your Network Load Balancer. For more information, see VPC flow logs in the *Amazon VPC User Guide*.

Create a flow log for each network interface for your load balancer. There is one network interface per load balancer subnet. To identify the network interfaces for a Network Load Balancer, look for the name of the load balancer in the description field of the network interface.

There are two entries for each connection through your Network Load Balancer, one for the frontend connection between the client and the load balancer and the other for the backend connection between the load balancer and the target. If the target is registered by instance ID, the connection appears to the instance as a connection from the client. If the security group of the instance doesn't allow connections from the client but the network ACLs for the load balancer subnet allow them, the logs for the network interface for the load balancer show "ACCEPT OK" for the frontend and backend connections, while the logs for the network interface for the instance show "REJECT OK" for the connection.

**Access logs**

You can use access logs to capture detailed information about TLS requests made to your load balancer. The log files are stored in Amazon S3. You can use these access logs to analyze traffic patterns and to troubleshoot issues with your targets. For more information, see Access logs for your Network Load Balancer (p. 58).

**CloudTrail logs**

You can use AWS CloudTrail to capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in Amazon S3. You can use these CloudTrail logs to determine which calls were made, the source IP address where the call came from, who made the call, when the call was made, and so on. For more information, see Logging API calls for your Network Load Balancer using AWS CloudTrail (p. 63).

# CloudWatch metrics for your Network Load Balancer

Elastic Load Balancing publishes data points to Amazon CloudWatch for your load balancers and your targets. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. Think of a metric as a variable to monitor, and the data points as the values of that variable over time. For example, you can monitor the total number of healthy targets for a

load balancer over a specified time period. Each data point has an associated time stamp and an optional unit of measurement.

You can use metrics to verify that your system is performing as expected. For example, you can create a CloudWatch alarm to monitor a specified metric and initiate an action (such as sending a notification to an email address) if the metric goes outside what you consider an acceptable range.

Elastic Load Balancing reports metrics to CloudWatch only when requests are flowing through the load balancer. If there are requests flowing through the load balancer, Elastic Load Balancing measures and sends its metrics in 60-second intervals. If there are no requests flowing through the load balancer or no data for a metric, the metric is not reported.

For more information, see the Amazon CloudWatch User Guide.

**Contents**



# Network Load Balancer metrics

The `AWS/NetworkELB` namespace includes the following metrics.

| Metric | Description |
|---|---|
| `ActiveFlowCount` | The total number of concurrent flows (or connections) from clients to targets. This metric includes connections in the SYN_SENT and ESTABLISHED states. TCP connections are not terminated at the load balancer, so a client opening a TCP connection to a target counts as a single flow.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistics are `Average`, `Maximum`, and `Minimum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`<br>• `AvailabilityZone`, `LoadBalancer` |
| `ActiveFlowCount_TCP` | The total number of concurrent TCP flows (or connections) from clients to targets. This metric includes only connections in the ESTABLISHED state. TCP connections are not terminated at the load balancer, so a client opening a TCP connection to a target counts as a single flow.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistics are `Average`, `Maximum`, and `Minimum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`<br>• `AvailabilityZone`, `LoadBalancer` |

| Metric | Description |
| --- | --- |
| `ActiveFlowCount_TLS` | The total number of concurrent TLS flows (or connections) from clients to targets. This metric includes only connections in the ESTABLISHED state.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistics are `Average`, `Maximum`, and `Minimum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`<br>• `AvailabilityZone`, `LoadBalancer` |
| `ActiveFlowCount_UDP` | The total number of concurrent UDP flows (or connections) from clients to targets.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistics are `Average`, `Maximum`, and `Minimum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`<br>• `AvailabilityZone`, `LoadBalancer` |
| `ClientTLSNegotiationErrorCount` | The total number of TLS handshakes that failed during negotiation between a client and a TLS listener.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is `Sum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`<br>• `AvailabilityZone`, `LoadBalancer` |
| `ConsumedLCUs` | The number of load balancer capacity units (LCU) used by your load balancer. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing.<br><br>**Reporting criteria**: Always reported<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• `LoadBalancer` |

| Metric | Description |
| --- | --- |
| `ConsumedLCUs_TCP` | The number of load balancer capacity units (LCU) used by your load balancer for TCP. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing.<br><br>**Reporting criteria**: Always reported<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• `LoadBalancer` |
| `ConsumedLCUs_TLS` | The number of load balancer capacity units (LCU) used by your load balancer for TLS. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing.<br><br>**Reporting criteria**: Always reported<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• `LoadBalancer` |
| `ConsumedLCUs_UDP` | The number of load balancer capacity units (LCU) used by your load balancer for UDP. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing.<br><br>**Reporting criteria**: Always reported<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• `LoadBalancer` |
| `HealthyHostCount` | The number of targets that are considered healthy.<br><br>**Reporting criteria**: Reported if health checks are enabled<br><br>**Statistics**: The most useful statistics are `Maximum` and `Minimum`.<br><br>**Dimensions**<br><br>• `LoadBalancer`, `TargetGroup`<br>• `AvailabilityZone`, `LoadBalancer`, `TargetGroup` |

| Metric | Description |
| --- | --- |
| NewFlowCount | The total number of new flows (or connections) established from clients to targets in the time period.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| NewFlowCount_TCP | The total number of new TCP flows (or connections) established from clients to targets in the time period.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| NewFlowCount_TLS | The total number of new TLS flows (or connections) established from clients to targets in the time period.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| NewFlowCount_UDP | The total number of new UDP flows (or connections) established from clients to targets in the time period.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |

| Metric | Description |
| --- | --- |
| ProcessedBytes | The total number of bytes processed by the load balancer, including TCP/IP headers. This count includes traffic to and from targets, minus health check traffic.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| ProcessedBytes_TCP | The total number of bytes processed by TCP listeners.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| ProcessedBytes_TLS | The total number of bytes processed by TLS listeners.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| ProcessedBytes_UDP | The total number of bytes processed by UDP listeners.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| TargetTLSNegotiationErrorCount | The total number of TLS handshakes that failed during negotiation between a TLS listener and a target.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |

| Metric | Description |
| --- | --- |
| TCP_Client_Reset_Count | The total number of reset (RST) packets sent from a client to a target. These resets are generated by the client and forwarded by the load balancer.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| TCP_ELB_Reset_Count | The total number of reset (RST) packets generated by the load balancer.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| TCP_Target_Reset_Count | The total number of reset (RST) packets sent from a target to a client. These resets are generated by the target and forwarded by the load balancer.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone, LoadBalancer |
| UnHealthyHostCount | The number of targets that are considered unhealthy.<br><br>**Reporting criteria**: Reported if health checks are enabled<br><br>**Statistics**: The most useful statistics are Maximum and Minimum.<br><br>**Dimensions**<br><br>• LoadBalancer, TargetGroup<br>• AvailabilityZone, LoadBalancer, TargetGroup |

# Metric dimensions for Network Load Balancers

To filter the metrics for your load balancer, use the following dimensions.

| Dimension | Description |
| --- | --- |
| `AvailabilityZone` | Filters the metric data by Availability Zone. |
| `LoadBalancer` | Filters the metric data by load balancer. Specify the load balancer as follows: net/*load-balancer-name*/*1234567890123456* (the final portion of the load balancer ARN). |
| `TargetGroup` | Filters the metric data by target group. Specify the target group as follows: targetgroup/*target-group-name*/*1234567890123456* (the final portion of the target group ARN). |

# Statistics for Network Load Balancer metrics

CloudWatch provides statistics based on the metric data points published by Elastic Load Balancing. Statistics are metric data aggregations over specified period of time. When you request statistics, the returned data stream is identified by the metric name and dimension. A dimension is a name/value pair that uniquely identifies a metric. For example, you can request statistics for all the healthy EC2 instances behind a load balancer launched in a specific Availability Zone.

The `Minimum` and `Maximum` statistics reflect the minimum and maximum values of the data points reported by the individual load balancer nodes in each sampling window. Increases in the maximum of `HealthyHostCount` correspond to decreases in the minimum of `UnHealthyHostCount`. Therefore, we recommend that you monitor your Network Load Balancer using either the maximum of `HealthyHostCount` or the minimum of `UnHealthyHostCount`.

The `Sum` statistic is the aggregate value across all load balancer nodes. Because metrics include multiple reports per period, `Sum` is only applicable to metrics that are aggregated across all load balancer nodes.

The `SampleCount` statistic is the number of samples measured. Because metrics are gathered based on sampling intervals and events, this statistic is typically not useful. For example, with `HealthyHostCount`, `SampleCount` is based on the number of samples that each load balancer node reports, not the number of healthy hosts.

# View CloudWatch metrics for your load balancer

You can view the CloudWatch metrics for your load balancers using the Amazon EC2 console. These metrics are displayed as monitoring graphs. The monitoring graphs show data points if the load balancer is active and receiving requests.

Alternatively, you can view metrics for your load balancer using the CloudWatch console.

**To view metrics using the Amazon EC2 console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. To view metrics filtered by target group, do the following:
   a. In the navigation pane, choose **Target Groups**.
   b. Select your target group and choose **Monitoring**.
   c. (Optional) To filter the results by time, select a time range from **Showing data for**.
   d. To get a larger view of a single metric, select its graph.
3. To view metrics filtered by load balancer, do the following:
   a. In the navigation pane, choose **Load Balancers**.

b. Select your load balancer and choose **Monitoring**.
c. (Optional) To filter the results by time, select a time range from **Showing data for**.
d. To get a larger view of a single metric, select its graph.

### To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **NetworkELB** namespace.
4. (Optional) To view a metric across all dimensions, type its name in the search field.

### To view metrics using the AWS CLI

Use the following list-metrics command to list the available metrics:

```
aws cloudwatch list-metrics --namespace AWS/NetworkELB
```

### To get the statistics for a metric using the AWS CLI

Use the following get-metric-statistics command get statistics for the specified metric and dimension. Note that CloudWatch treats each unique combination of dimensions as a separate metric. You can't retrieve statistics using combinations of dimensions that were not specially published. You must specify the same dimensions that were used when the metrics were created.

```
aws cloudwatch get-metric-statistics --namespace AWS/NetworkELB \
--metric-name UnHealthyHostCount --statistics Average  --period 3600 \
--dimensions Name=LoadBalancer,Value=net/my-load-balancer/50dc6c495c0c9188 \
Name=TargetGroup,Value=targetgroup/my-targets/73e2d6bc24d8a067 \
--start-time 2017-04-18T00:00:00Z --end-time 2017-04-21T00:00:00Z
```

The following is example output:

```
{
    "Datapoints": [
        {
            "Timestamp": "2017-04-18T22:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2017-04-18T04:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        ...
    ],
    "Label": "UnHealthyHostCount"
}
```

# Access logs for your Network Load Balancer

Elastic Load Balancing provides access logs that capture detailed information about the TLS requests sent to your Network Load Balancer. You can use these access logs to analyze traffic patterns and troubleshoot issues.

**Important**
Access logs are created only if the load balancer has a TLS listener and they contain information only about TLS requests.

Access logging is an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logging for your load balancer, Elastic Load Balancing captures the logs as compressed files and stores them in the Amazon S3 bucket that you specify. You can disable access logging at any time.

If you enable server-side encryption with Amazon S3-managed encryption keys (SSE-S3) for your S3 bucket, each access log file is automatically encrypted before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action as there is no difference in the way you access encrypted or unencrypted log files. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated. For more information, see Protecting data using server-side encryption with Amazon S3-managed encryption keys (SSE-S3) in the *Amazon Simple Storage Service Developer Guide*.

There is no additional charge for access logs. You are charged storage costs for Amazon S3, but not charged for the bandwidth used by Elastic Load Balancing to send log files to Amazon S3. For more information about storage costs, see Amazon S3 Pricing.

# Access log files

Elastic Load Balancing publishes a log file for each load balancer node every 5 minutes. Log delivery is eventually consistent. The load balancer can deliver multiple logs for the same period. This usually happens if the site has high traffic.

The file names of the access logs use the following format:

```
bucket[/prefix]/AWSLogs/aws-account-id/elasticloadbalancing/region/yyyy/mm/dd/aws-account-id_elasticloadbalancing_region_load-balancer-id_end-time_random-string.log.gz
```

*bucket*

The name of the S3 bucket.

*prefix*

The prefix (logical hierarchy) in the bucket. If you don't specify a prefix, the logs are placed at the root level of the bucket.

*aws-account-id*

The AWS account ID of the owner.

*region*

The region for your load balancer and S3 bucket.

*yyyy/mm/dd*

The date that the log was delivered.

*load-balancer-id*

The resource ID of the load balancer. If the resource ID contains any forward slashes (/), they are replaced with periods (.).

*end-time*

The date and time that the logging interval ended. For example, an end time of 20181220T2340Z contains entries for requests made between 23:35 and 23:40.

*random-string*

A system-generated random string.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. For more information, see Object lifecycle management in the *Amazon Simple Storage Service Developer Guide*.

# Access log entries

The following table describes the fields of an access log entry, in order. All fields are delimited by spaces. When new fields are introduced, they are added to the end of the log entry. When processing the log files, you should ignore any fields at the end of the log entry that you were not expecting.

| Field | Description |
|---|---|
| type | The type of listener. The supported value is `tls`. |
| version | The version of the log entry. The current version is 2.0. |
| time | The time recorded at the end of the TLS connection, in ISO 8601 format. |
| elb | The resource ID of the load balancer. |
| listener | The resource ID of the TLS listener for the connection. |
| client:port | The IP address and port of the client. |
| destination:port | The IP address and port of the destination. If the client connects directly to the load balancer, the destination is the listener. If the client connects using a VPC endpoint service, the destination is the VPC endpoint. |
| connection_time | The total time for the connection to complete, from start to closure, in milliseconds. |
| tls_handshake_time | The total time for the TLS handshake to complete after the TCP connection is established, including client-side delays, in milliseconds. This time is included in the connection_time field. |
| received_bytes | The count of bytes received by the load balancer from the client, after decryption. |
| sent_bytes | The count of bytes sent by the load balancer to the client, before encryption. |
| incoming_tls_alert | The integer value of TLS alerts received by the load balancer from the client, if present. Otherwise, this value is set to -. |
| chosen_cert_arn | The ARN of the certificate served to the client. If no valid client hello message is sent, this value is set to -. |
| chosen_cert_serial | Reserved for future use. This value is always set to -. |
| tls_cipher | The cipher suite negotiated with the client, in OpenSSL format. If TLS negotiation does not complete, this value is set to -. |
| tls_protocol_version | The TLS protocol negotiated with the client, in string format. The possible values are `tlsv10`, `tlsv11`, and `tlsv12`. If TLS negotiation does not complete, this value is set to -. |

| Field | Description |
|---|---|
| tls_named_group | Reserved for future use. This value is always set to -. |
| domain_name | The value of the server_name extension in the client hello message. This value is URL-encoded. If no valid client hello message is sent or the extension is not present, this value is set to -. |
| alpn_fe_protocol | The application protocol negotiated with the client, in string format. The possible values are `h2`, `http/1.1`, and `http/1.0`. If no ALPN policy is configured in the TLS listener, no matching protocol is found, or no valid protocol list is sent, this value is set to -. |
| alpn_be_protocol | The application protocol negotiated with the target, in string format. The possible values are `h2`, `http/1.1`, and `http/1.0`. If no ALPN policy is configured in the TLS listener, no matching protocol is found, or no valid protocol list is sent, this value is set to -. |
| alpn_client_preference_list | The value of the application_layer_protocol_negotiation extension in the client hello message. This value is URL-encoded. Each protocol is enclosed in double quotes and protocols are separated by a comma. If no ALPN policy is configured in the TLS listener, no valid client hello message is sent, or the extension is not present, this value is set to -. The string is truncated if it is longer than 256 bytes. |

## Example log entries

The following are example log entries. Note that the text appears on multiple lines only to make it easier to read.

The following is an example for a TLS listener without an ALPN policy.

```
tls 2.0 2018-12-20T02:59:40 net/my-network-loadbalancer/c6e77e28c25b2234 g3d4b5e8bb8464cd
72.21.218.154:51341 172.100.100.185:443 5 2 98 246 -
arn:aws:acm:us-east-2:671290407336:certificate/2a108f19-aded-46b0-8493-c63eb1ef4a99 -
ECDHE-RSA-AES128-SHA tlsv12 -
my-network-loadbalancer-c6e77e28c25b2234.elb.us-east-2.amazonaws.com
- - -
```

The following is an example for a TLS listener with an ALPN policy.

```
tls 2.0 2020-04-01T08:51:42 net/my-network-loadbalancer/c6e77e28c25b2234 g3d4b5e8bb8464cd
72.21.218.154:51341 172.100.100.185:443 5 2 98 246 -
arn:aws:acm:us-east-2:671290407336:certificate/2a108f19-aded-46b0-8493-c63eb1ef4a99 -
ECDHE-RSA-AES128-SHA tlsv12 -
my-network-loadbalancer-c6e77e28c25b2234.elb.us-east-2.amazonaws.com
h2 h2 "h2","http/1.1"
```

# Bucket requirements

When you enable access logging, you must specify an S3 bucket for the access logs. The bucket can be owned by a different account than the account that owns the load balancer. The bucket must meet the following requirements.

**Requirements**

- The bucket must be located in the same region as the load balancer.

- The prefix that you specify must not include AWSLogs. We add the portion of the file name starting with AWSLogs after the bucket name and prefix that you specify.
- Amazon S3-Managed Encryption Keys (SSE-S3) is required. No other encryption options are supported.
- The bucket must have a bucket policy that grants permission to write the access logs to your bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. The following is an example policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket_name/prefix/AWSLogs/aws-account-id/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::bucket_name"
    }
  ]
}
```

# Enable access logging

When you enable access logging for your load balancer, you must specify the name of the S3 bucket where the load balancer will store the logs. For more information, see Bucket requirements (p. 61).

### To enable access logging using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select your load balancer.
4. On the **Description** tab, choose **Edit attributes**.
5. On the **Edit load balancer attributes** page, do the following:
   a. For **Access logs**, choose **Enable**.
   b. For **S3 location**, type the name of your S3 bucket, including any prefix (for example, `my-loadbalancer-logs/my-app`). You can specify the name of an existing bucket or a name for a new bucket. If you specify an existing bucket, be sure that you own this bucket and that you configured the required bucket policy.
   c. (Optional) If the bucket does not exist, choose **Create this location for me**. You must specify a name that is unique across all existing bucket names in Amazon S3 and follows the DNS naming

conventions. For more information, see Rules for bucket naming in the *Amazon Simple Storage Service Developer Guide*.

d. Choose **Save**.

**To enable access logging using the AWS CLI**

Use the modify-load-balancer-attributes command.

## Disable access logging

You can disable access logging for your load balancer at any time. After you disable access logging, your access logs remain in your S3 bucket until you delete the them. For more information, see Working with buckets in the *Amazon Simple Storage Service Console User Guide*.

**To disable access logging using the console**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select your load balancer.
4. On the **Description** tab, choose **Edit attributes**.
5. For **Access logs**, clear **Enable**.
6. Choose **Save**.

**To disable access logging using the AWS CLI**

Use the modify-load-balancer-attributes command.

## Processing access log files

The access log files are compressed. If you open the files using the Amazon S3 console, they are uncompressed and the information is displayed. If you download the files, you must uncompress them to view the information.

If there is a lot of demand on your website, your load balancer can generate log files with gigabytes of data. You might not be able to process such a large amount of data using line-by-line processing. Therefore, you might have to use analytical tools that provide parallel processing solutions. For example, you can use the following analytical tools to analyze and process access logs:

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. For more information, see Querying Network Load Balancer logs in the *Amazon Athena User Guide*.
- Loggly
- Splunk
- Sumo Logic

# Logging API calls for your Network Load Balancer using AWS CloudTrail

Elastic Load Balancing is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Elastic Load Balancing. CloudTrail captures all API calls for Elastic

Load Balancing as events. The calls captured include calls from the AWS Management Console and code calls to the Elastic Load Balancing API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Elastic Load Balancing. 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 Elastic Load Balancing, 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.

# Elastic Load Balancing information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Elastic Load Balancing, 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 Elastic Load Balancing, 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 Elastic Load Balancing actions for Network Load Balancers are logged by CloudTrail and are documented in the Elastic Load Balancing API Reference version 2015-12-01. For example, calls to the `CreateLoadBalancer` and `DeleteLoadBalancer` 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 Elastic Load Balancing 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 log files include events for all AWS API calls for your AWS account, not just Elastic Load Balancing API calls. You can locate calls to the Elastic Load Balancing API by checking for `eventSource` elements

with the value `elasticloadbalancing.amazonaws.com`. To view a record for a specific action, such as `CreateLoadBalancer`, check for `eventName` elements with the action name.

The following are example CloudTrail log records for Elastic Load Balancing for a user who created a Network Load Balancer and then deleted it using the AWS CLI. You can identify the CLI using the `userAgent` elements. You can identify the requested API calls using the `eventName` elements. Information about the user (`Alice`) can be found in the `userIdentity` element.

**Example Example: CreateLoadBalancer**

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-01T15:31:48Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "CreateLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "subnets": ["subnet-8360a9e7","subnet-b7d581c0"],
        "securityGroups": ["sg-5943793c"],
        "name": "my-load-balancer",
        "scheme": "internet-facing",
        "type": "network"
    },
    "responseElements": {
        "loadBalancers":[{
            "type": "network",
            "ipAddressType": "ipv4",
            "loadBalancerName": "my-load-balancer",
            "vpcId": "vpc-3ac0fb5f",
            "securityGroups": ["sg-5943793c"],
            "state": {"code":"provisioning"},
            "availabilityZones": [
               {"subnetId":"subnet-8360a9e7","zoneName":"us-west-2a"},
               {"subnetId":"subnet-b7d581c0","zoneName":"us-west-2b"}
            ],
            "dNSName": "my-load-balancer-1836718677.us-west-2.elb.amazonaws.com",
            "canonicalHostedZoneId": "Z2P70J7HTTTPLU",
            "createdTime": "Apr 11, 2016 5:23:50 PM",
            "loadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-load-balancer/ffcddace1759e1d0",
            "scheme": "internet-facing"
        }]
    },
    "requestID": "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
    "eventID": "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE",
    "eventType": "AwsApiCall",
    "apiVersion": "2015-12-01",
    "recipientAccountId": "123456789012"
}
```

**Example Example: DeleteLoadBalancer**

```
{
```

```
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-01T15:31:48Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "DeleteLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "loadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-load-balancer/ffcddace1759e1d0"
    },
    "responseElements": null,
    "requestID": "349598b3-000e-11e6-a82b-298133eEXAMPLE",
    "eventID": "75e81c95-4012-421f-a0cf-babdaEXAMPLE",
    "eventType": "AwsApiCall",
    "apiVersion": "2015-12-01",
    "recipientAccountId": "123456789012"
}
```

# Troubleshoot your Network Load Balancer

The following information can help you troubleshoot issues with your Network Load Balancer.

## A registered target is not in service

If a target is taking longer than expected to enter the `InService` state, it might be failing health checks. Your target is not in service until it passes one health check. For more information, see Health checks for your target groups (p. 38).

Verify that your instance is failing health checks and then check for the following:

**A security group does not allow traffic**

The security groups associated with an instance must allow traffic from the load balancer using the health check port and health check protocol. For more information, see Target security groups (p. 43).

**A network access control list (ACL) does not allow traffic**

The network ACL associated with the subnets for your instances and the subnets for your load balancer must allow traffic and health checks from the load balancer. For more information, see Network ACLs (p. 44).

## Requests are not routed to targets

Check for the following:

**A security group does not allow traffic**

The security groups associated with the instances must allow traffic on the listener port from client IP addresses (if targets are specified by instance ID) or load balancer nodes (if targets are specified by IP address). For more information, see Target security groups (p. 43).

**A network access control list (ACL) does not allow traffic**

The network ACLs associated with the subnets for your VPC must allow the load balancer and targets to communicate in both directions on the listener port. For more information, see Network ACLs (p. 44).

**The targets are in an Availability Zone that is not enabled**

If you register targets in an Availability Zone but do not enable the Availability Zone, these registered targets do not receive traffic from the load balancer.

**The instance is in a peered VPC**

If you have instances in a VPC that is peered with the load balancer VPC, you must register them with your load balancer by IP address, not by instance ID.

# Targets receive more health check requests than expected

Health checks for a Network Load Balancer are distributed and use a consensus mechanism to determine target health. Therefore, targets receive more than the number of health checks configured through the `HealthCheckIntervalSeconds` setting.

# Targets receive fewer health check requests than expected

Check whether `net.ipv4.tcp_tw_recycle` is enabled. This setting is known to cause issues with load balancers. The `net.ipv4.tcp_tw_reuse` setting is considered a safer alternative.

# Unhealthy targets receive requests from the load balancer

If there is at least one healthy registered target for your load balancer, the load balancer routes requests only to its healthy registered targets. If there are only unhealthy registered targets, the load balancer routes requests to all registered targets.

# Target fails HTTP or HTTPS health checks due to host header mismatch

The HTTP host header in the health check request contains the IP address of the load balancer node and the listener port, not the IP address of the target and the health check port. If you are mapping incoming requests by host header, you must ensure that health checks match any HTTP host header. Another option is to add a separate HTTP service on a different port and configure the target group to use that port for health checks instead. Alternatively, consider using TCP health checks.

# Connections time out for requests from a target to its load balancer

Check whether you have an internal load balancer with targets registered by instance ID. Internal load balancers do not support hairpinning or loopback. When you register targets by instance ID, the source IP addresses of clients are preserved. If an instance is a client of an internal load balancer that it's registered with by instance ID, the connection succeeds only if the request is routed to a different instance. Otherwise, the source and destination IP addresses are the same and the connection times out.

If an instance must send requests to a load balancer that it's registered with, do one of the following:

- Register instances by IP address instead of instance ID. When using Amazon Elastic Container Service, use the `awsvpc` network mode with your tasks to ensure that target groups require registration by IP address.

- Ensure that containers that must communicate are on different container instances.
- Use an Internet-facing load balancer.

# Performance decreases when moving targets to a Network Load Balancer

Both Classic Load Balancers and Application Load Balancers use connection multiplexing, but Network Load Balancers do not. Therefore, your targets can receive more TCP connections behind a Network Load Balancer. Be sure that your targets are prepared to handle the volume of connection requests they might receive.

# Port allocation errors connecting through AWS PrivateLink

If your Network Load Balancer is associated with a VPC endpoint service, it supports 55,000 simultaneous connections or about 55,000 connections per minute to each unique target (IP address and port). If you exceed these connections, there is an increased chance of port allocation errors. To fix the port allocation errors, add more targets to the target group.

# Quotas for your Network Load Balancers

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for your Network Load Balancers, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **Elastic Load Balancing**. You can also use the describe-account-limits (AWS CLI) command for Elastic Load Balancing.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the Elastic Load Balancing limit increase form.

Your AWS account has the following quotas related to Network Load Balancers.

### Regional

- Network Load Balancers per Region: 50
- Network Load Balancer ENIs per Amazon VPC: 300 *
- Target groups per Region: 3,000 **

* *Each Network Load Balancer uses one ENI per zone.*

** *This quota is shared by target groups for your Application Load Balancers and Network Load Balancers.*

### Load balancer

- Listeners per load balancer: 50
- Targets per load balancer: 3,000
- Subnets per Availability Zone per load balancer: 1
- [Cross-zone load balancing disabled] Targets per Availability Zone per load balancer: 500
- [Cross-zone load balancing enabled] Targets per load balancer: 500
- Certificates per load balancer (not counting default certificates): 25

### Target group

- Load balancers per target group: 1
- Targets per target group: 1,000

# Document history for Network Load Balancers

The following table describes the releases for Network Load Balancers.

| update-history-change | update-history-description | update-history-date |
| --- | --- | --- |
| Security policy for FS supporting TLS version 1.2 | This release adds a security policy for Forward Secrecy (FS) supporting TLS version 1.2. | November 24, 2020 |
| Dual-stack mode | This release adds supports for dual-stack mode, which enables clients to connect to the load balancer using both IPv4 addresses and IPv6 addresses. | November 13, 2020 |
| Connection termination on deregistration (p. 71) | This release adds support to close connections to deregistered targets after the end of the deregistration timeout. | November 13, 2020 |
| ALPN policies | This release adds support for Application-Layer Protocol Negotiation (ALPN) preference lists. | May 27, 2020 |
| Sticky sessions | This release adds support for sticky sessions based on source IP address and protocol. | February 28, 2020 |
| Shared subnets (p. 71) | This release adds support for specifying subnets that were shared with you by another AWS account. | November 26, 2019 |
| Private IP addresses (p. 71) | This release enables you to provide a private IP address from the IPv4 address range of the subnet you specify when you enable an Availability Zone for an internal load balancer. | November 25, 2019 |
| Add subnets (p. 71) | This release adds support for enabling additional Availability Zones after you create your load balancer. | November 25, 2019 |
| SNI support | This release adds support for Server Name Indication (SNI). | September 12, 2019 |
| UDP protocol (p. 71) | This release adds support for the UDP protocol. | June 24, 2019 |

| | | |
|---|---|---|
| TLS protocol | This release adds support for the TLS protocol. | January 24, 2019 |
| Cross-zone load balancing (p. 71) | This release adds support for enabling cross-zone load balancing. | February 22, 2018 |
| Proxy protocol | This release adds support for enabling Proxy Protocol. | November 17, 2017 |
| IP addresses as targets | This release adds support for registering IP addresses as targets. | September 21, 2017 |
| New load balancer type (p. 71) | This release of Elastic Load Balancing introduces Network Load Balancers. | September 7, 2017 |