aws

Network Load Balancers

# Elastic Load Balancing

# 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.

A *target group* routes requests to one or more registered targets, such as EC2 instances, using the protocol and the port number that you specify. Network Load Balancer target groups support the TCP, UDP, TCP_UDP, and TLS protocols. 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
- Listeners
- Target groups

# 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 request from a client, it selects a target from the target group for the default rule. It attempts to send the request to the selected target using the protocol and port that you specified.

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 Update the Availability Zones for your Network Load Balancer.

To increase the fault tolerance of your applications, you can enable multiple Availability Zones for your load balancer and ensure that each target group has at least one target in each enabled Availability Zone. 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 how you register targets. For example, you can register instance IDs, IP addresses, or an Application Load Balancer. The target type also affects whether the client IP addresses are preserved. For more information, see the section called “Client IP preservation”.

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.

# Getting started

To create a Network Load Balancer using the AWS Management Console, AWS CLI, or AWS CloudFormation, see Create a Network Load Balancer.

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

# Pricing

For more information, see Elastic Load Balancing pricing.

# Network Load Balancers

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

To configure your Network Load Balancer, you create target groups, and then register targets with your target groups. Your Network Load Balancer is most effective if you ensure that each enabled Availability Zone has at least one registered target. You also create listeners 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 Network Load Balancer can be in one of the following states:

`provisioning`

The Network Load Balancer is being set up.

`active`

The Network Load Balancer is fully set up and ready to route traffic.

`failed`

The Network Load Balancer couldn't be set up.

# IP address type

You can set the types of IP addresses that clients can use with your Network Load Balancer.

Network Load Balancers support the following IP address types:

**`ipv4`**

Clients must connect using IPv4 addresses (for example, 192.0.2.1).

**`dualstack`**

Clients can connect to the Network 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).

**Considerations**

- The Network Load Balancer communicates with targets based on the IP address type of the target group.
- To support source IP preservation for UDP IPv6 listeners, ensure that **Enable prefix for IPv6 source NAT** is turned on.
- When you enable dualstack mode for the Network Load Balancer, Elastic Load Balancing provides an AAAA DNS record for the Network Load Balancer. Clients that communicate with the Network Load Balancer using IPv4 addresses resolve the A DNS record. Clients that communicate with the Network Load Balancer using IPv6 addresses resolve the AAAA DNS record.

- Access to your internal dualstack Network Load Balancer through the internet gateway is blocked to prevent unintended internet access. However, this does not prevent other internet access (for example, through peering, Transit Gateway, AWS Direct Connect, or AWS VPN).

For more information, see Update the IP address types for your Network Load Balancer.

## 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 no longer tracked. If a client or target sends data after the idle timeout period elapses, the client receives a TCP RST packet to indicate that the connection is no longer valid.

The default idle timeout value for TCP flows is 350 seconds, but can be updated to any value between 60-6000 seconds. Clients or targets can use TCP keepalive packets to restart the idle timeout. Keepalive packets sent to maintain TLS connections can't contain data or payload.

The connection idle timeout for TLS listeners is 350 seconds and can't be modified. When a TLS listener receives a TCP keepalive packet from either a client or a target, the load balancer generates TCP keepalive packets and sends them to both the front-end and back-end connections every 20 seconds. You can't modify this behavior.

While UDP is connectionless, the load balancer maintains UDP flow state based on the source and destination IP addresses and ports. This ensures 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. This cannot be changed.

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

For more information, see Update idle timeout.

## Load balancer attributes

You can configure your Network Load Balancer by editing its attributes. For more information, see Edit load balancer attributes.

The following are the load balancer attributes for Network Load Balancers:

`access_logs.s3.enabled`

Indicates whether access logs stored in Amazon S3 are enabled. The default is `false`.

`access_logs.s3.bucket`

The name of the Amazon S3 bucket for the access logs. This attribute is required if access logs are enabled. For more information, see Bucket requirements.

`access_logs.s3.prefix`

The prefix for the location in the Amazon S3 bucket.

`deletion_protection.enabled`

Indicates whether deletion protection is enabled. The default is `false`.

`ipv6.deny_all_igw_traffic`

Blocks internet gateway (IGW) access to the Network Load Balancer, preventing unintended access to your internal Network Load Balancer through an internet gateway. It is set to `false` for internet-facing Network Load Balancers and `true` for internal Network Load Balancers. This attribute does not prevent non-IGW internet access (for example, through peering, Transit Gateway, AWS Direct Connect, or AWS VPN).

`load_balancing.cross_zone.enabled`

Indicates whether cross-zone load balancing is enabled. The default is `false`.

`dns_record.client_routing_policy`

Indicates how traffic is distributed among the Network Load Balancers Availability Zones. The possible values are `availability_zone_affinity` with 100 percent zonal affinity, `partial_availability_zone_affinity` with 85 percent zonal affinity, and `any_availability_zone` with 0 percent zonal affinity.

`secondary_ips.auto_assigned.per_subnet`

The number of secondary IP addresses to configure. Use to resolve port allocation errors if you can't add targets. The valid range is 0 to 7. The default is 0. After you set this value, you can't decrease it.

`zonal_shift.config.enabled`

Indicates whether zonal shift is enabled. The default is `false`.

# Cross-zone load balancing

By default, each Network Load Balancer node distributes traffic across the registered targets in its Availability Zone only. If you turn on cross-zone load balancing, each Network Load Balancer node distributes traffic across the registered targets in all enabled Availability Zones. You can also turn on cross-zone load balancing at the target group level. For more information, see the section called "Cross-zone load balancing" and Cross-zone load balancing in the *Elastic Load Balancing User Guide*.

# 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 Network Load Balancer. When a client makes a request using this custom domain name, the DNS server resolves it to the DNS name for your Network 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 DNS record to route requests to your Network Load Balancer. For more information, see the documentation for your DNS service. For example, if you use Amazon Route 53 as your DNS service, you create an alias record that points to your Network Load Balancer. For more information, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.

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

**Linux or Mac**

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

**Windows**

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

The Network Load Balancer has DNS records for its nodes. You can use DNS names with the following syntax to determine the IP addresses of the Network 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
```

# Load balancer zonal health

Network Load Balancers have zonal DNS records and IP addresses in Route 53 for each enabled availability zone. When a Network Load Balancer fails a zonal health check for a particular availability zone, its DNS record is removed from Route 53. Load balancer zonal health is monitored using the Amazon CloudWatch metric `ZonalHealthStatus`, giving you more insight into events that cause a fail-away to implement preventative measure to ensure optimal application availability. For more information see, Network Load Balancer metrics.

Network Load Balancers can fail zonal health checks for multiple reasons, causing them to become unhealthy. See below for common causes of unhealthy Network Load Balancers caused by failed zonal health checks.

**Check for the following possible causes:**

- There are no healthy targets for the load balancer
- The number of healthy targets is less than the configured minimum
- There is a zonal shift or zonal auto-shift in progress
- Traffic is being automatically shifted to healthy zones due to detected issues

# Create a Network Load Balancer

A Network Load Balancer takes requests from clients and distributes them across targets in a target group, such as EC2 instances. For more information, see the the section called “Network Load Balancer overview”.

**Tasks**



## Prerequisites

- Decide which Availability Zones and IP address types your application will support. Configure your load balancer VPC with subnets in each of these Availability Zones. If the application will support both IPv4 and IPv6 traffic, ensure that the subnets have both IPv4 and IPv6 CIDRs. Deploy at least one target in each Availability Zone.
- Ensure that the security groups for target instances 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 the section called “Target security groups”.
- Ensure that the security groups for target instances allow traffic from the load balancer on the health check port using the health check protocol.

## Create the load balancer

As part of creating a Network Load Balancer, you'll create the load balancer, at least one listener, and at least one target group. Your load balancer is ready to handle client requests when there is at least one healthy registered target in each of its enabled Availability Zones.

Console

**To create a Network Load Balancer**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.

3. Choose **Create load balancer**.
4. Under **Network Load Balancer**, choose **Create**.
5. **Basic configuration**

   a. For **Load balancer name**, enter a name for your Network Load Balancer. The name must be unique within your set of load balancers in the Region. It can have a maximum of 32 characters, and contain only alphanumeric characters and hyphens. It must not begin or end with a hyphen, or with `internal-`.

   b. For **Scheme**, choose **Internet-facing** or **Internal**. An internet-facing Network Load Balancer routes requests from clients to targets over the internet. An internal Network Load Balancer routes requests to targets using private IP addresses.

   c. For **Load balancer IP address type**, choose **IPv4** if your clients use IPv4 addresses to communicate with the Network Load Balancer or **Dualstack** if your clients use both IPv4 and IPv6 addresses to communicate with the Network Load Balancer.

6. **Network mapping**

   a. For **VPC**, select the VPC that you prepared for your load balancer. With an internet-facing load balancer, only VPCs with an internet gateway are available for selection.

   b. With a dualstack load balancer, you can't add a UDP listener unless **Enable prefix for IPv6 source NAT** is **On (source NAT prefixes per subnet)**.

   c. For **Availability Zones and subnets**, select at least one Availability Zone, and select one subnet per zone. Note that subnets that were shared with you are available for selection.

      If you select multiple Availability Zones and ensure that you have registered targets in each selected zone, this increases the fault tolerance of your application.

   d. With 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.

      With an internal load balancer, you can enter a private IPv4 address from the address range of each subnet or let AWS select one for you.

      With a dualstack load balancer, you can enter an IPv6 address from the address range of each subnet or let AWS select one for you.

      For a load balancer with source NAT enabled, you can enter a custom IPv6 prefix or let AWS select one for you.

7. **Security groups**

   We preselect the default security group for the load balancer VPC. You can select additional security groups as needed. If you don't have a security group that meets your needs, choose **create a new security group** to create one now. For more information, see Create a security group in the *Amazon VPC User Guide*.

   > **Warning**
   >
   > If you don't associate any security groups with your Network Load Balancer now, you can't associate them later on.

8. **Listeners and routing**

   a. The default is a listener that accepts TCP traffic on port 80. You can keep the default listener settings, or modify **Protocol** and **Port** as needed.

   b. For **Default action**, select a target group to forward traffic. If you don't have a target group that meets your needs, choose **Create target group** to create one now. For more information, see Create a target group.

   c. (Optional) Choose **Add listener tag** and enter a tag key and a tag value.

   d. (Optional) Choose **Add listener** to add another listener (for example, a TLS listener).

9. **Secure listener settings**

   This section appears only if you add a TLS listener.

   a. For **Security policy**, choose a security policy that meets your requirements. For more information, see Security policies.

   b. For **Default SSL/TLS server certificate**, choose **From ACM** as the certificate source. Select a certificate that you provisioned or imported using AWS Certificate Manager. If you don't have an available certificate in ACM but do have a certificate for use with your load balancer, select **Import certificate** and provide the required information. Otherwise, choose **Request new ACM certificate**. For more information, see AWS Certificate Manager certificates in the *AWS Certificate Manager User Guide*.

   c. (Optional) For **ALPN policy**, choose a policy to enable ALPN. For more information, see the section called “ALPN policies”.

10. **Load balancer tags**

(Optional) Expand **Load balancer tags**. Choose **Add new tag** and enter a tag key and a tag value. For more information, see Tags.

11. **Summary**

    Review your configuration, and choose **Create load balancer**. A few default attributes are applied to your Network Load Balancer during creation. You can view and edit them after creating the Network Load Balancer. For more information, see Load balancer attributes.

AWS CLI

**To create a Network Load Balancer**

Use the create-load-balancer command.

The following example creates an internet-facing load balancer with two enabled Availability Zones and a security group.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --type network \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890 \
    --security-groups sg-1111222233334444
```

**To create an internal Network Load Balancer**

Include the `--scheme` option as shown in the following example.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --type network \
    --scheme internal \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890 \
    --security-groups sg-1111222233334444
```

**To create a dualstack Network Load Balancer**

Include the `--ip-address-type` option as shown in the following example.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
```

```
    --type network \
    --ip-address-type dualstack \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890 \
    --security-groups sg-1111222233334444
```

**To add a listener**

Use the create-listener command. For examples, see Create a listener.

CloudFormation

**To create a Network Load Balancer**

Define a resource of type AWS::ElasticLoadBalancingV2::LoadBalancer.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      IpAddressType: dualstack
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      Tags:
        - Key: 'department'
          Value: '123'
```

**To add a listener**

Define a resource of type AWS::ElasticLoadBalancingV2::Listener. For examples, see Create a listener.

## Test the load balancer

After creating your Network Load Balancer, you can verify that your EC2 instances have passed the initial health check, and then test that the Network Load Balancer is sending traffic to your EC2 instances. To delete the Network Load Balancer, see Delete a Network Load Balancer.

**To test the Network Load Balancer**

1. After the Network Load Balancer is created, choose **Close**.
2. In the left navigation pane, choose **Target Groups**.
3. Select the new target group.
4. 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 Network Load Balancer. For more information, see Target health status.
5. In the navigation pane, choose **Load Balancers**.
6. Select the new Network Load Balancer.
7. Copy the DNS name of the Network 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.

## Next steps

After you create your load balancer, you might want to do the following:

- Configure load balancer attributes.
- Configure target group attributes.
- [TLS listeners] Add certificates to the optional certificate list.
- Configure monitoring features.

# Update the Availability Zones for your Network Load Balancer

You can enable or disable the Availability Zones for your Network Load Balancer at any time. When you enable an Availability Zone, you must specify one subnet from that Availability 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. Enabling multiple Availability Zones helps improve the fault tolerance of your applications.

Elastic Load Balancing creates a Network Load Balancer node in the Availability Zone you choose, and a network interface for the selected subnet in that Availability Zone. Each Network Load Balancer node in the Availability Zone uses the network interface to get an IPv4 address. You can view these network interfaces, but they can't be modified.

**Considerations**

- For internet-facing Network Load Balancers, the subnets that you specify must have at least 8 available IP addresses. For internal Network 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. However, you can specify a subnet in a non-constrained Availability Zone 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.
- You can't remove a subnet if the Network Load Balancer has active Amazon VPC endpoint associations.
- When adding back a previously removed subnet, a new network interface is created with a different ID.
- Subnet changes within the same Availability Zone must be independent actions. You first complete removing the existing subnet, then you can add the new subnet.
- Subnet removal can take up to 3 minutes to complete.

When creating an internet-facing Network Load Balancer, you can choose to specify an Elastic IP address for each Availability Zone. Elastic IP addresses provide your Network Load Balancer with static IP addresses. If you choose not to specify an Elastic IP address, AWS will assign one Elastic IP address for each Availability Zone.

When creating an internal Network Load Balancer, you can choose to specify a private IP address from each subnet. Private IP addresses provide your Network Load Balancer with static IP addresses. If you choose not to specify a private IP address, AWS assigns one for you.

Before updating the Availability Zones for your Network Load Balancer, we recommend you evaluate for any potential impact on existing connections, traffic flows, or production workloads.

**⚠ Updating an Availability Zone can be disruptive**

- When a subnet is removed, its associated Elastic Network Interface (ENI) is deleted. This causes all active connections in the Availability Zone to be terminated.
- After a subnet is removed, all targets within the Availability Zone it was associated with are marked as `unused`. This results in those targets being removed from the available target pool, and all active connections to those targets being terminated. This includes any connections originating from other Availability Zones when utilizing cross-zone load balancing.
- Network Load Balancers have a 60 second Time To Live (TTL) for their Fully Qualified Domain Name (FQDN). When an Availability Zone that contains active targets is removed any existing client connections may experience timeouts until DNS resolution occurs again, and traffic is shifted to any remaining Availability Zones.

Console

**To modify the Availability Zones**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Network mapping** tab, choose **Edit subnets**.
5. To enable an Availability Zone, select its check box and select one subnet. If there is only one available subnet, it is selected for you.
6. To change the subnet for an enabled Availability Zone, choose one of the other subnets from the list.
7. To disable an Availability Zone, clear its check box.
8. Choose **Save changes**.

AWS CLI

**To modify the Availability Zones**

Use the set-subnets command.

```
aws elbv2 set-subnets \
    --load-balancer-arn load-balancer-arn \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890
```

CloudFormation

**To modify the Availability Zones**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref new-subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
```

# Update the IP address types for your Network Load Balancer

You can configure your Network Load Balancer so that clients can communicate with the Network Load Balancer using IPv4 addresses only, or using both IPv4 and IPv6 addresses (dualstack). The Network Load Balancer communicates with targets based on the IP address type of the target group. For more information, see IP address type.

**Dualstack requirements**

- You can set the IP address type when you create the Network Load Balancer and update it at any time.
- The virtual private cloud (VPC) and subnets that you specify for the Network Load Balancer must have associated IPv6 CIDR blocks. For more information, see IPv6 addresses in the *Amazon EC2 User Guide*.
- The route tables for the Network Load Balancer subnets must route IPv6 traffic.
- The network ACLs for the Network Load Balancer subnets must allow IPv6 traffic.

Console

**To update the IP address type**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the check box for the Network 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 changes**.

AWS CLI

**To update the IP address type**

Use the set-ip-address-type command.

```
aws elbv2 set-ip-address-type \
    --load-balancer-arn load-balancer-arn \
    --ip-address-type dualstack
```

CloudFormation

**To update the IP address type**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      IpAddressType: dualstack
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
```

```
      SecurityGroups:
        - !Ref mySecurityGroup
```

# Edit attributes for your Network Load Balancer

After you create a Network Load Balancer, you can edit its attributes.

**Load balancer attributes**

- Deletion protection
- Cross-zone load balancing
- Availability Zone DNS affinity
- Secondary IP addresses

## Deletion protection

To prevent your Network Load Balancer from being deleted accidentally, you can enable deletion protection. By default, deletion protection is disabled for your Network Load Balancer.

If you enable deletion protection for your Network Load Balancer, you must disable it before you can delete the Network Load Balancer.

Console

**To enable or disable deletion protection**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the name of the Network Load Balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Protection**, enable or disable **Deletion protection**.
6. Choose **Save changes**.

AWS CLI

**To enable or disable deletion protection**

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

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=deletion_protection.enabled,Value=true"
```

CloudFormation

**To enable or disable deletion protection**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `deletion_protection.enabled` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "deletion_protection.enabled"
          Value: "true"
```

## Cross-zone load balancing

With Network Load Balancers, cross-zone load balancing is off by default at the load balancer level, but you can turn it on at any time. For target groups, the default is to use the load balancer setting, but you can override the default by explicitly turning cross-zone load balancing on or off at the target group level. For more information, see the section called “Cross-zone load balancing”.

Console

**To enable or disable cross-zone load balancing for a 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 name of the load balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit load balancer attributes** page, turn **Cross-zone load balancing** on or off.
6. Choose **Save changes**.

AWS CLI

**To enable or disable cross-zone load balancing for a load balancer**

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

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=load_balancing.cross_zone.enabled,Value=true"
```

CloudFormation

**To enable or disable cross-zone load balancing for a load balancer**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `load_balancing.cross_zone.enabled` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
```

```
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "load_balancing.cross_zone.enabled"
          Value: "true"
```

## Availability Zone DNS affinity

When using the default client routing policy, requests sent to your Network Load Balancers DNS name will receive any healthy Network Load Balancer IP addresses. This leads to the distribution of client connections across the Network Load Balancer's Availability Zones. With the Availability Zone affinity routing policies, client DNS queries favor Network Load Balancer IP addresses in their own Availability Zone. This helps improve both latency and resiliency, as clients do not need to cross Availability Zone boundaries when connecting to targets.

Availability Zone affinity routing policies only apply to clients resolving the Network Load Balancers DNS name using Route 53 Resolver. For more information, see What is Amazon Route 53 Resolver? in the *Amazon Route 53 Developer Guide*

**Client routing policies available to Network Load Balancers using Route 53 resolver:**

- **Availability Zone affinity** – *100 percent zonal affinity*

  Client DNS queries will favor Network Load Balancer IP address in their own Availability Zone. Queries may resolve to other zones if there are no healthy Network Load Balancer IP addresses in their own zone.
- **Partial Availability Zone affinity** – *85 percent zonal affinity*

  85 percent of client DNS queries will favor Network Load Balancer IP addresses in their own Availability Zone, while the remaining queries resolve to any healthy zone. Queries may resolve to other healthy zones if there are no healthy IPs in their zone. When there are no healthy IPs in any zone, queries resolve to any zone.
- **Any Availability Zone** (default) – *0 percent zonal affinity*

  Client DNS queries are resolved among healthy Network Load Balancer IP addresses across all Network Load Balancer Availability Zones.

Availability Zone affinity helps route requests from the client to the Network Load Balancer, while cross-zone load balancing is used to help route requests from the Network Load Balancer to the

targets. When using Availability Zone affinity, cross-zone load balancing should be turned off, this ensures the Network Load Balancer traffic from clients to targets remains within the same Availability Zone. With this configuration, client traffic is sent to the same Network Load Balancer Availability Zone, so it's recommended to configure your application to scale independently in each Availability Zone. This is an important consideration when the number of clients per Availability zone, or the traffic per Availability Zone are not the same. For more information, see Cross-zone load balancing for target groups.

When an Availability Zone is considered unhealthy, or when a zonal shift is started, the zonal IP address will be considered unhealthy and not returned to clients unless fail open is in effect. Availability Zone affinity is maintained when the DNS record fails open. This helps keep Availability Zones independent and prevent potential cross zone failures.

When using Availability Zone affinity, times of imbalance between Availability Zones are expected. It's recommended ensuring your targets are scaling at the zonal level, to support each Availability Zones workload. In cases where these imbalances are significant, it's recommended turning off Availability Zone affinity. This allows even distribution of client connections between all the Network Load Balancer's Availability Zones within 60 seconds, or the DNS TTL.

**Before using Availability Zone affinity, consider the following:**

- Availability Zone affinity causes changes on all of the Network Load Balancers clients who are using Route 53 Resolver.
  - Clients aren't able to decide between zonal-local and multi-zone DNS resolutions. Availability Zone affinity decides for them.
  - Clients aren't provided with a reliable method to determine when they're being impacted by Availability Zone affinity, or how to know which IP address is in which Availability Zone.
- When using Availability Zone affinity with Network Load Balancers and Route 53 Resolver, we recommend clients use the Route 53 Resolver inbound endpoint in their own Availability Zone.
- Clients will remain assigned to their zone-local IP address until it is deemed fully unhealthy according to DNS health checks, and is removed from DNS.
- Using Availability Zone affinity with cross-zone load balancing on can lead to unbalanced distribution of client connections between Availability Zones. It's recommended to configure your application stack to scale independently in each Availability Zone, ensuring it can support zonal clients traffic.
- If cross-zone load balancing is on, the Network Load Balancer is subject to cross zone impact.

- The load on each of the Network Load Balancers Availability Zones will be proportional to the zonal locations of clients requests. If you don't configure how many clients are running in which Availability Zone, you will have to independently scale each Availability Zone reactively.

## Monitoring

It is recommended to track the distribution of connections between Availability Zones, using the zonal Network Load Balancer metrics. You can use metrics to view the number of new and active connections per zone.

We recommend tracking the following:

- **`ActiveFlowCount`** – The total number of concurrent flows (or connections) from clients to targets.
- **`NewFlowCount`** – The total number of new flows (or connections) established from clients to targets in the time period.
- **`HealthyHostCount`** – The number of targets that are considered healthy.
- **`UnHealthyHostCount`** – The number of targets that are considered unhealthy.

For more information, see CloudWatch metrics for your Network Load Balancer

## Enable Availability Zone affinity

Console

**To enable Availability Zone affinity**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the name of the Network Load Balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Availability Zone routing configuration**, **Client routing policy (DNS record)**, select **Availability Zone affinity** or **Partial Availability Zone affinity**.
6. Choose **Save changes**.

AWS CLI

**To enable Availability Zone affinity**

Use the modify-load-balancer-attributes command with the `dns_record.client_routing_policy` attribute.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes
 "Key=dns_record.client_routing_policy,Value=partial_availability_zone_affinity"
```

CloudFormation

**To enable Availability Zone affinity**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `dns_record.client_routing_policy` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "dns_record.client_routing_policy"
          Value: "partial_availability_zone_affinity"
```

## Secondary IP addresses

If you experience port allocation errors and you can't add targets to the target group to resolve them, you can add secondary IP addresses to the load balancer network interfaces. For each zone where the load balancer is enabled, we select IPv4 addresses from the load balancer subnet and assign them to the corresponding network interface. These secondary IP addresses are used to

establish connections with targets. They are also used for health check traffic. We recommend that you add one secondary IP address to start with, monitor the `PortAllocationErrors` metric, and add another secondary IP address only if the port allocation errors are not resolved.

> **Warning**
>
> After you add secondary IP addresses, you can't remove them. The only way to release the secondary IP addresses is to delete the load balancer. Before you add secondary IP addresses, verify that there are enough available IPv4 addresses in the load balancer subnets.

Console

**To add a secondary IP address**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the name of the Network Load Balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. Expand **Special case attributes**, unlock the **Secondary IP addresses auto assigned per subnet** attribute, and choose the number of secondary IP addresses.
6. Choose **Save changes**.

AWS CLI

**To add a secondary IP address**

Use the modify-load-balancer-attributes command with the `secondary_ips.auto_assigned.per_subnet` attribute.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=secondary_ips.auto_assigned.per_subnet,Value=1"
```

You can use the describe-network-interfaces command to get the IPv4 addresses for the load balancer network interfaces. The `--filters` parameter scopes the results to the network interfaces for Network Load Balancers and the `--query` parameter further scopes the results to

the load balancer with the specified name and displays only the specified fields. You can include additional fields as needed.

```
aws elbv2 describe-network-interfaces \
    --filters "Name=interface-type,Values=network_load_balancer" \
    --query "NetworkInterfaces[?contains(Description,'my-nlb')].
{ID:NetworkInterfaceId,AZ:AvailabilityZone,Addresses:PrivateIpAddresses[*]}"
```

CloudFormation

**To add a secondary IP address**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `secondary_ips.auto_assigned.per_subnet` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "secondary_ips.auto_assigned.per_subnet"
          Value: "1"
```

# Update the security groups for your Network Load Balancer

You can associate a security group with your Network Load Balancer to control the traffic that is allowed to reach and leave the Network Load Balancer. You specify the ports, protocols, and sources to allow for inbound traffic and the ports, protocols, and destinations to allow for outbound traffic. If you don't assign a security group to your Network Load Balancer, all client traffic can reach the Network Load Balancer listeners and all traffic can leave the Network Load Balancer.

You can add a rule to the security groups associated with your targets that references the security group associated with your Network Load Balancer. This allows clients to send traffic to your targets through your Network Load Balancer, but prevents them from sending traffic directly to your targets. Referencing the security group associated with your Network Load Balancer in the security groups associated with your targets ensures that your targets accept traffic from your Network Load Balancer even if you enable client IP preservation for your Network Load Balancer.

You are not charged for traffic that is blocked by inbound security group rules.

**Contents**



## Considerations

- You can associate security groups with a Network Load Balancer when you create it. If you create a Network Load Balancer without associating any security groups, you can't associate them with the Network Load Balancer later on. We recommend that you associate a security group with your Network Load Balancer when you create it.
- After you create a Network Load Balancer with associated security groups, you can change the security groups associated with the Network Load Balancer at any time.
- Health checks are subject to outbound rules, but not inbound rules. You must ensure that outbound rules don't block health check traffic. Otherwise, the Network Load Balancer considers the targets unhealthy.
- You can control whether PrivateLink traffic is subject to inbound rules. If you enable inbound rules on PrivateLink traffic, the source of the traffic is the private IP address of the client, not the endpoint interface.

## Example: Filter client traffic

The following inbound rules in the security group associated with your Network Load Balancer allow only traffic that comes from the specified address range. If this is an internal Network Load Balancer, you can specify a VPC CIDR range as the source to allow only traffic from a specific VPC. If this is an internet-facing Network Load Balancer that must accept traffic from anywhere on the internet, you can specify 0.0.0.0/0 as the source.

**Inbound**

| Protocol | Source | Port range | Comment |
|---|---|---|---|
| *protocol* | *client IP address range* | *listener port* | Allows inbound traffic from the source CIDR on the listener port |
| ICMP | 0.0.0.0/0 | All | Allows inbound ICMP traffic to support MTU or Path MTU Discovery † |

† For more information, see Path MTU Discovery in the *Amazon EC2 User Guide*.

**Outbound**

| Protocol | Destination | Port range | Comment |
|---|---|---|---|
| All | Anywhere | All | Allows all outbound traffic |

## Example: Accept traffic only from the Network Load Balancer

Suppose that your Network Load Balancer has a security group sg-111112222233333. Use the following rules in the security groups associated with your target instances to ensure that they accept traffic only from the Network Load Balancer. You must ensure that the targets accept traffic from the Network Load Balancer on both the target port and the health check port. For more information, see the section called “Target security groups”.

**Inbound**

| Protocol | Source | Port range | Comment |
|---|---|---|---|
| *protocol* | sg-111112 222233333 | *target port* | Allows inbound traffic from the Network Load Balancer on the target port |
| *protocol* | sg-111112 222233333 | *health check* | Allows inbound traffic from the Network Load Balancer on the health check port |

**Outbound**

| Protocol | Destination | Port range | Comment |
|---|---|---|---|
| All | Anywhere | Any | Allows all outbound traffic |

## Update the associated security groups

If you associated at least one security group with a Network Load Balancer when you created it, you can update the security groups for that Network Load Balancer at any time.

Console

**To update the security groups**

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 Network Load Balancer.
4. On the **Security** tab, choose **Edit**.
5. To associate a security group with your Network Load Balancer, select it. To remove a security group from your Network Load Balancer, clear it.
6. Choose **Save changes**.

AWS CLI

**To update the security groups**

Use the set-security-groups command.

```
aws elbv2 set-security-groups \
    --load-balancer-arn load-balancer-arn \
    --security-groups sg-1234567890abcdef0 sg-0abcdef0123456789
```

CloudFormation

**To update the security groups**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
        - !Ref myNewSecurityGroup
```

## Update the security settings

By default, we apply the inbound security group rules to all traffic sent to the Network Load Balancer. However, you might not want to apply these rules to traffic sent to the Network Load Balancer through AWS PrivateLink, which can originate from overlapping IP addresses. In this case, you can configure the Network Load Balancer so that we do not apply the inbound rules for traffic sent to the Network Load Balancer through AWS PrivateLink.

Console

**To update the security settings**

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 Network Load Balancer.
4. On the **Security** tab, choose **Edit**.
5. Under **Security setting**, clear **Enforce inbound rules on PrivateLink traffic**.
6. Choose **Save changes**.

AWS CLI

**To update the security settings**

Use the set-security-groups command.

```
aws elbv2 set-security-groups \
    --load-balancer-arn load-balancer-arn \
    --enforce-security-group-inbound-rules-on-private-link-traffic off
```

CloudFormation

**To update the security settings**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      EnforceSecurityGroupInboundRulesOnPrivateLinkTraffic: off
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
```

## Monitor Network Load Balancer security groups

Use the `SecurityGroupBlockedFlowCount_Inbound` and `SecurityGroupBlockedFlowCount_Outbound` CloudWatch metrics to monitor the count of flows that are blocked by the Network Load Balancer security groups. Blocked traffic is not reflected in other metrics. For more information, see the section called “CloudWatch metrics”.

Use VPC flow logs to monitor traffic that is accepted or rejected by the Network Load Balancer security groups. For more information, see VPC flow logs in the *Amazon VPC User Guide*.

# Tag a Network Load Balancer

Tags help you to categorize your Network Load Balancers in different ways. For example, you can tag a resource by purpose, owner, or environment.

You can add multiple tags to each Network Load Balancer. If you add a tag with a key that is already associated with the Network Load Balancer, it updates the value of that tag.

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

Console

**To update the tags for a load balancer**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.

3. Select the check box for the Network Load Balancer.
4. On the **Tags** tab, choose **Manage tags**.
5. To add a tag, choose **Add tag** and enter the tag key and tag value. Allowed characters are letters, spaces, numbers (in UTF-8), and the following special characters: + - = . _ : / @. Do not use leading or trailing spaces. Tag values are case-sensitive.
6. To update a tag, enter new values in **Key** or **Value**.
7. To delete a tag, choose **Remove** next to the tag.
8. Choose **Save changes**.

AWS CLI

**To add tags**

Use the add-tags command. The following example adds two tags.

```
aws elbv2 add-tags \
    --resource-arns load-balancer-arn \
    --tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

**To remove tags**

Use the remove-tags command. The following example removes the tags with the specified keys.

```
aws elbv2 remove-tags \
    --resource-arns load-balancer-arn \
    --tag-keys project department
```

CloudFormation

**To add tags**

Define a resource of type AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the Tags property.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
```

```
  Properties:
    Name: my-nlb
    Type: network
    Scheme: internal
    Subnets:
      - !Ref subnet-AZ1
      - !Ref subnet-AZ2
    SecurityGroups:
      - !Ref mySecurityGroup
    Tags:
      - Key: 'project'
        Value: 'lima'
      - Key: 'department'
        Value: 'digital-media'
```

# Delete a Network Load Balancer

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

You can't delete a Network Load Balancer if deletion protection is enabled. For more information, see Deletion protection.

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

Deleting a Network Load Balancer also deletes its listeners. Deleting a Network 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 for your Network Load Balancer.

Console

**To delete a Network Load Balancer**

1. If you have a DNS record for your domain that points to your Network Load Balancer, point it to a new location and wait for the DNS change to take effect before deleting your Network Load Balancer. For example:

- If the record is a CNAME record with a Time To Live (TTL) of 300 seconds, wait at least 300 seconds before continuing to the next step.
- If the record is a Route 53 Alias(A) record, wait at least 60 seconds.
- If using Route 53, the record change takes 60 seconds to propagate to all global Route 53 name servers. Add this time to the TTL value of the record that is being updated.

2. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
3. In the navigation pane, choose **Load Balancers**.
4. Select the check box for the Network Load Balancer.
5. Choose **Actions**, **Delete load balancer**.
6. When prompted for confirmation, enter **`confirm`** and choose **Delete**.

AWS CLI

**To delete a Network Load Balancer**

Use the delete-load-balancer command.

```
aws elbv2 delete-load-balancer \
    --load-balancer-arn load-balancer-arn
```

# View the Network Load Balancer resource map

The Network Load Balancer resource map provides an interactive display of your Network Load Balancers architecture, including its associated listeners, target groups, and targets. The resource map also highlights the relationships and routing paths between all resources, producing a visual representation of your Network Load Balancers configuration.

**To view the resource map for your load balancer**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the Network Load Balancer.
4. Choose the **Resource map** tab.

# Resource map components

### Map views

There are two views available in the Network Load Balancer resource map: **Overview**, and **Unhealthy Target Map**. **Overview** is selected by default and displays all of your Network Load Balancer's resources. Selecting the **Unhealthy Target Map** view will only display the unhealthy targets and the resources associated to them.

The **Unhealthy Target Map** view can be used to troubleshoot targets that are failing health checks. For more information, see Troubleshoot unhealthy targets using the resource map.

### Resource columns

The Network Load Balancer resource map contains three resource columns, one for each resource type. The resource groups are **Listeners**, **Target groups**, and **Targets**.

### Resource tiles

Each resource within a column has its own tile, which displays details about that specific resource.

- Hovering over a resource tile highlights the relationships between it and other resources.
- Selecting a resource tile highlights the relationships between it and other resources, and displays additional details about that resource.
  - **target group health summary:** The number of registered targets for each health status.
  - **target health status:** The target's current health status and description.

> **Note**
>
> You can turn off **Show resource details** to hide additional details within the resource map.

- Each resource tile contains a link that, when selected, navigates to that resource's details page.
  - **Listeners** - Select the listeners protocol:port. For example, `TCP:80`
  - **Target groups** - Select the target group name. For example, `my-target-group`
  - **Targets** - Select the targets ID. For example, `i-1234567890abcdef0`

### Export the resource map

Selecting **Export** gives you the option of exporting the current view of your Network Load Balancer's resource map as a PDF.

# Zonal shift for your Network Load Balancer

Zonal shift is a capability in Amazon Application Recovery Controller (ARC). With zonal shift, you can shift a Network Load Balancer resource away from an impaired Availability Zone with a single action. This way, you can continue operating from other healthy Availability Zones in an AWS Region.

When you start a zonal shift, your Network Load Balancer stops routing traffic to targets in the affected Availability Zone. Existing connections to targets in the affected Availability Zone are not terminated by zonal shift. It might take several minutes for these connections to complete gracefully.

**Contents**



## Before you begin a zonal shift

- Zonal shift is disabled by default and must be enabled on each Network Load Balancer. For more information, see Enable zonal shift for your Network Load Balancer.
- You can start a zonal shift for a specific Network Load Balancer only for a single Availability Zone. You can't start a zonal shift for multiple Availability Zones.
- AWS proactively removes zonal Network Load Balancer IP addresses from DNS when multiple infrastructure issues impact services. Always check current Availability Zone capacity before you start a zonal shift. If you use a zonal shift on your Network Load Balancer, the Availability Zone affected by the zonal shift also loses target capacity.
- During zonal shift on Network Load Balancers with cross-zone load balancing enabled, the zonal load balancer IP addresses are removed from DNS. Existing connections to targets in the

impaired Availability Zone persist until they organically close, while new connections are no longer routed to targets in the impaired Availability Zone.

For more information, see Best practices for zonal shifts in ARC in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

## Zonal shift administrative override

Targets that belong to a Network Load Balancer will include a new status `AdministrativeOverride`, which is independent from the `TargetHealth` state.

When a zonal shift is started for a Network Load Balancer, all targets within the zone being shifted away from are considered administratively overridden. The Network Load Balancer stops routing new traffic to administratively overridden targets. Existing connections remain intact until they are organically closed.

The possible `AdministrativeOverride` states are:

**unknown**

State cannot be propagated due to an internal error

**no_override**

No override is currently active on target

**zonal_shift_active**

Zonal shift is active in target Availability Zone

**zonal_shift_delegated_to_dns**

This target's zonal shift state is not available through DescribeTargetHealth but can be viewed directly through the Amazon ARC API or console

## Enable zonal shift for your Network Load Balancer

Zonal shift is disabled by default and must be enabled on each Network Load Balancer. This ensures that you can start a zonal shift using only the specific Network Load Balancers that you want. For more information, see the section called “Zonal shift”.

### Prerequisites

If you enable cross-zone load balancing for the load balancer, every target group attached to the load balancer must meet the following requirements before you can enable zonal shift.

- The target group protocol must be `TCP` or `TLS`.
- The target group type must not be `alb`.
- Connection termination for unhealthy targets must be disabled.
- The `load_balancing.cross_zone.enabled` target group attribute must be `true` or `use_load_balancer_configuration` (the default).

Console

**To enable zonal shift**

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 Network Load Balancer.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Availability Zone routing configuration**, for **ARC zonal shift integration**, choose **Enable**.
6. Choose **Save changes**.

AWS CLI

**To enable zonal shift**

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

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=zonal_shift.config.enabled,Value=true"
```

CloudFormation

**To enable zonal shift**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `zonal_shift.config.enabled` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        -Key: "zonal_shift.config.enabled"
         Value: "true"
```

## Start a zonal shift for your Network Load Balancer

Zonal shift in ARC enables you to temporarily move traffic for supported resources away from an Availability Zone so that your application can continue to operate normally with other Availability Zones in an AWS Region.

**Prerequisite**

Before you begin, verify that you enabled zonal shift for the load balancer.

Console

This procedure explains how to start a zonal shift using the Amazon EC2 console. For steps to start a zonal shift using the ARC console, see Starting a zonal shift in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

**To start a zonal shift**

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 Network Load Balancer.

4. On the **Integrations** tab, expand **Amazon Application Recovery Controller (ARC)** and choose **Start zonal shift**.
5. Select the Availability Zone that you want to move traffic away from.
6. Choose or enter an expiration for the zonal shift. A zonal shift can initially be set from 1 minute up to three days (72 hours).

   All zonal shifts are temporary. You must set an expiration, but you can update active shifts later to set a new expiration.
7. Enter a comment. You can update the zonal shift later to edit the comment.
8. Select the check box to acknowledge that starting a zonal shift reduces capacity for your application by shifting traffic away from the Availability Zone.
9. Choose **Confirm**.

AWS CLI

**To start a zonal shift**

Use the Amazon Application Recovery Controller (ARC) start-zonal-shift command.

```
aws arc-zonal-shift start-zonal-shift \
    --resource-identifier load-balancer-arn \
    --away-from use2-az2 \
    --expires-in 2h \
    --comment "zonal shift due to scheduled maintenance"
```

## Update a zonal shift for your Network Load Balancer

You can update a zonal shift to set a new expiration, or edit or replace the comment for the zonal shift.

Console

This procedure explains how to update a zonal shift using the Amazon EC2 console. For steps to update a zonal shift using the Amazon Application Recovery Controller (ARC) console, see Updating a zonal shift in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

**To update a zonal shift**

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 an Application Load Balancer with an active zonal shift.
4. On the **Integrations** tab, expand **Amazon Application Recovery Controller (ARC)** and choose **Update zonal shift**.

   This opens the ARC console to continue the update process.
5. (Optional) For **Set zonal shift expiration**, select or enter an expiration.
6. (Optional) For **Comment**, optionally edit the existing comment or enter a new comment.
7. Choose **Update**.

AWS CLI

**To update a zonal shift**

Use the Amazon Application Recovery Controller (ARC) update-zonal-shift command.

```
aws arc-zonal-shift update-zonal-shift \
    --zonal-shift-id 9ac9ec1e-1df1-0755-3dc5-8cf57EXAMPLE \
    --expires-in 1h \
    --comment "extending zonal shift for scheduled maintenance"
```

## Cancel a zonal shift for your Network Load Balancer

You can cancel a zonal shift any time before it expires. You can cancel zonal shifts that you initiate, or zonal shifts that AWS starts for a resource for a practice run for zonal autoshift.

Console

This procedure explains how to cancel a zonal shift using the Amazon EC2 console. For steps to cancel a zonal shift using the Amazon Application Recovery Controller (ARC) console, see Canceling a zonal shift in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

**To cancel a zonal shift**

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 a Network Load Balancer with an active zonal shift.
4. On the **Integrations** tab, under **Amazon Application Recovery Controller (ARC)**, choose **Cancel zonal shift**.

   This opens the ARC console to continue the cancelation process.
5. Choose **Cancel zonal shift**.
6. When prompted for confirmation, choose **Confirm**.

AWS CLI

**To cancel a zonal shift**

Use the Amazon Application Recovery Controller (ARC) cancel-zonal-shift command.

```
aws arc-zonal-shift cancel-zonal-shift \
    --zonal-shift-id 9ac9ec1e-1df1-0755-3dc5-8cf57EXAMPLE
```

# Capacity reservations for your Network Load Balancer

Load balancer Capacity Unit (LCU) reservations allow you to reserve a static minimum capacity for your load balancer. Network Load Balancers automatically scale to support detected workloads and meet capacity needs. When minimum capacity is configured, your load balancer continues scaling up or down based on the traffic received, but also prevents the capacity from going lower than the minimum capacity configured.

Consider using LCU reservation in following situations:

- You have an upcoming event that will have a sudden, unusual high traffic and want to ensure your load balancer can support the sudden traffic spike during the event.
- You have unpredictable spiky traffic due to the nature of your workload for a short period.
- You are setting up your load balancer to on-board or migrate your services at a specific start time and need start with a high capacity instead of waiting for auto-scaling to take effect.
- You are migrating workloads between load balancers and want to configure the destination to match the scale of the source.

### Estimate the capacity that you need

When determining the amount of capacity you should reserve for your load balancer, we recommend performing load testing or reviewing historical workload data that represents the upcoming traffic you expect. Using the Elastic Load Balancing console, you can estimate how much capacity you need to reserve based on the reviewed traffic.

Alternatively, you can refer to CloudWatch metric **ProcessedBytes** to determine the right level of capacity. Capacity for your load balancer is reserved in LCUs, with each LCU being equal to 2.2Mbps. You can use the Max (**ProcessedBytes**) metric to see the maximum per-minute throughput traffic on the load balancer, then convert that throughput to LCUs using a conversion rate of 2.2Mbps equals 1 LCU.

If you don't have historical workload data to reference and cannot perform load testing, you can estimate capacity needed using the LCU reservation calculator. The LCU reservation calculator uses data based on historical workloads AWS observe and may not represent your specific workload. For more information, see Load Balancer Capacity Unit Reservation Calculator.

### Supported Regions

This feature is available only in the following Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Asia Pacific (Hong Kong)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (Stockholm)

### Quotas for LCU reservations

Your account has quotas related to LCUs. For more information, see the section called “Load Balancer Capacity Units”.

# Request Load balancer Capacity Unit reservation for your Network Load Balancer

Before you use LCU reservation, review the following:

- LCU reservation is not supported on Network Load Balancers using TLS listeners.
- LCU reservation only supports reserving throughput capacity for Network Load Balancers. When requesting a LCU reservation, convert your capacity needs from Mbps to LCUs using the conversion rate of 1 LCU to 2.2 Mbps.
- Capacity is reserved at the regional level and is evenly distributed across availability zones. Confirm you have enough evenly distributed targets in each availability zone before turning on LCU reservation.
- LCU reservation requests are fulfilled on a first come first serve basis, and depends on available capacity for a zone at that time. Most requests are typically fulfilled within an hour, but can take up to a few hours.
- To update an existing reservation, the previous request must be provisioned or failed. You can increase reserved capacity as many times as you need, however you can only decrease the reserved capacity two times per day.
- You will continue to incur charges for any reserved or provisioned capacity until they are terminated or cancelled.

Console

**To request an LCU reservation**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer name.
4. On the **Capacity** tab, choose **Edit LCU Reservation**.
5. Select **Historic reference based estimate**.
6. Select the reference period to view the recommended reserved LCU level.
7. If you do not have historic reference workload, you can choose **Manual estimate** and enter the number of LCUs to be reserved.
8. Choose **Save**.

AWS CLI

**To request an LCU reservation**

Use the modify-capacity-reservation command.

```
aws elbv2 modify-capacity-reservation \
    --load-balancer-arn load-balancer-arn \
    --minimum-load-balancer-capacity CapacityUnits=3000
```

CloudFormation

**To request an LCU reservation**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      MinimumLoadBalancerCapacity:
        CapacityUnits: 3000
```

## Update or cancel Load Balancer Capacity Unit reservations for your Network Load Balancer

If the traffic patterns for your load balancer change, you can update or cancel the LCU reservation for your load balancer.

Console

**To update or cancel an LCU reservation**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer name.
4. On the **Capacity** tab, do one of the following:
   a. To update the LCU reservation choose **Edit LCU Reservation**.
   b. To cancel the LCU reservation, choose **Cancel Capacity**.

AWS CLI

**To cancel an LCU reservation**

Use the modify-capacity-reservation command.

```
aws elbv2 modify-capacity-reservation \
    --load-balancer-arn load-balancer-arn \
    --reset-capacity-reservation
```

## Monitor Load balancer Capacity Unit reservation for your Network Load Balancer

### Reservation status

The following are the possible status values for an LCU reservation:

- pending - Indicates the reservation it is in the process of provisioning.
- provisioned - Indicates the reserved capacity is ready and available to use.
- failed - Indicates the request cannot be completed at the time.
- rebalancing - Indicates an availability zone has been added or removed and the load balancer is rebalancing capacity.

### LCU utilization

To determine reserved LCU utilization, you can compare the per-minute `ProcessedBytes` metric with the per-hour Sum(ReservedLCUs). To convert bytes per minute to LCU per hour, use (bytes per min)*8/60/ (10^6)/2.2.

Console

**To view the status of an LCU reservation**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer name.
4. On the **Capacity** tab, you can view the **Reservation Status** and **Reserved LCU** value.

AWS CLI

**To monitor the status of an LCU reservation**

Use the describe-capacity-reservation command.

```
aws elbv2 describe-capacity-reservation \
    --load-balancer-arn load-balancer-arn
```

# Listeners for your Network Load Balancers

A *listener* is a process that checks for connection requests, using the protocol and port that you configure. Before you start using your Network Load Balancer, you must add at least one listener. If your load balancer has no listeners, it can't receive traffic from clients. The rule that you define for a listener determines how the load balancer routes requests to the targets that you register, such as EC2 instances.

**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 at least one SSL server certificate on the listener. For more information, see Server certificates.

If you must ensure that the targets decrypt TLS traffic instead of the load balancer, you can create a TCP listener on port 443 instead of creating a TLS listener. With a TCP listener, the load balancer passes encrypted traffic through to the targets without decrypting it.

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.

A UDP listener for a dualstack load balancer requires IPv6 target groups.

WebSockets is supported only on TCP, TLS, and TCP_UDP 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).

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

## Listener attributes

The following are the listener attributes for Network Load Balancers:

`tcp.idle_timeout.seconds`

The tcp idle timeout value, in seconds. The valid range is 60-6000 seconds. The default is 350 seconds.

For more information, see Update idle timeout.

## 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.

# Secure listeners

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. Note that if you need to pass encrypted traffic to the targets without the load balancer decrypting it, create a TCP listener on port 443 instead of creating a TLS listener. The load balancer passes the request to the target as is, without decrypting it.

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 mutual TLS authentication (mTLS). For mTLS support, create a TCP listener instead of a TLS listener. The load balancer passes the request through as is, so you can implement mTLS on the target.

Network Load Balancers support TLS resumption using PSK for TLS 1.3, and session tickets for TLS 1.2 and older. Resumptions with session ID, or when multiple certificates are configured in the listener using SNI, are not supported. The 0-RTT data feature and early_data extension are not implemented.

For related demos, see TLS Support on Network Load Balancer and SNI Support on Network Load Balancer.

# 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.

**Supported ALPN Policies**

The following are the supported ALPN policies:

`HTTP10nly`

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

`HTTP20nly`

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

`HTTP20ptional`

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 and Update the ALPN policy.

# 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.

- 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 for your Network Load Balancer.
- You can't use an IPv4 target group with a UDP listener for a `dualstack` load balancer.

## 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.

Console

**To add a listener**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the name of the load balancer to open its details page.
4. On the **Listeners** tab, choose **Add listener**.
5. For **Protocol**, choose **TCP**, **UDP**, **TCP_UDP**, or **TLS**. Keep the default port or type a different port.
6. For **Default action**, choose an available target group. If you don't have a target group that meets your needs, choose **Create target group** to create one now. For more information, see Create a target group.
7. [TLS listeners] For **Security policy**, we recommend that you keep the default security policy.
8. [TLS listeners] For **Default SSL/TLS server certificate**, choose the default certificate. You can select the certificate from one of the following sources:

   - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM**, then choose the certificate from **Certificate (from ACM)**.
   - If you imported a certificate using IAM, choose **From IAM**, and then choose the certificate from **Certificate (from IAM)**.
   - If you have a certificate, choose **Import certificate**. Choose either **Import to ACM** or **Import to IAM**. For **Certificate private key**, copy and paste the contents of the private key file (PEM-encoded). For **Certificate body**, copy and paste the contents of the public

key certificate file (PEM-encoded). For **Certificate Chain**, copy and paste the contents of the certificate chain file (PEM-encoded), unless you are using a self-signed certificate and it's not important that browsers implicitly accept the certificate.

9. [TLS listeners] For **ALPN policy**, choose a policy to enable ALPN or choose **None** to disable ALPN. For more information, see ALPN policies.
10. Choose **Add**.
11. [TLS listeners] To add certificates to the optional certificate list, see Add certificates to the certificate list.

AWS CLI

**To create a target group**

If you don't have a target group that you can use for the default action, use the create-target-group command to create one now. For examples, see Create a target group.

**To add a TCP listener**

Use the create-listener command, specifying the TCP protocol.

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

**To add a TLS listener**

Use the create-listener command specifying the TLS protocol.

```
aws elbv2 create-listener \
    --load-balancer-arn load-balancer-arn \
    --protocol TLS \
    --port 443 \
    --certificates CertificateArn=certificate-arn \
    --ssl-policy ELBSecurityPolicy-TLS13-1-2-Res-2021-06 \
    --default-actions Type=forward,TargetGroupArn=target-group-arn
```

**To add a UDP listener**

Use the create-listener command specifying the UDP protocol.

```
aws elbv2 create-listener \
    --load-balancer-arn load-balancer-arn \
    --protocol UDP \
    --port 53 \
    --default-actions Type=forward,TargetGroupArn=target-group-arn
```

CloudFormation

**To add a TCP listener**

Define a resource of type AWS::ElasticLoadBalancingV2::Listener using the TCP protocol.

```
Resources:
  myTCPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TCP
      Port: 80
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

**To add a TLS listener**

Define a resource of type AWS::ElasticLoadBalancingV2::Listener using the TLS protocol.

```
Resources:
  myTLSListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TLS
      Port: 443
      SslPolicy: "ELBSecurityPolicy-TLS13-1-2-Res-2021-06"
      Certificates:
        - CertificateArn: "certificate-arn"
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

**To add a UDP listener**

Define a resource of type AWS::ElasticLoadBalancingV2::Listener using the UDP protocol.

```
Resources:
  myUDPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: UDP
      Port: 53
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

## Server certificates for your Network Load Balancer

When you create a secure listener for your Network Load Balancer, you must deploy at least one certificate on the load balancer. 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. The domain name on the certificate must match the custom domain name record so that we can verify the TLS connection. If they do not match, the traffic is not encrypted.

You must specify a fully qualified domain name (FQDN) for your certificate, such as `www.example.com` or an apex domain name such as `example.com`. You can also use an asterisk (*) as a wild card to protect several site names in the same domain. When you request a wild-card certificate, the asterisk (*) must be in the leftmost position of the domain name and can protect only one subdomain level. For instance, `*.example.com` protects `corp.example.com`, and `images.example.com`, but it cannot protect `test.login.example.com`. Also note that `*.example.com` protects only the subdomains of `example.com`, it does not protect the bare or apex domain (`example.com`). The wild-card name appears in the **Subject** field and in the **Subject Alternative Name** extension of the certificate. For more information about public certificates, see Requesting a public certificate in the *AWS Certificate Manager User Guide*.

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*.

## Supported key algorithms

- RSA 1024-bit
- RSA 2048-bit
- RSA 3072-bit
- ECDSA 256-bit
- ECDSA 384-bit
- ECDSA 521-bit

## Default certificate

When you create a TLS listener, you must specify at least 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.

If you specify additional certificates in a certificate list, 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.

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.

### 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 for your Network Load Balancer

When you create a TLS listener, you must select a security policy. A security policy determines which ciphers and protocols are supported during SSL negotiations between your load balancer

and clients. You can update the security policy for your load balancer if your requirements change or when we release a new security policy. For more information, see Update the security policy.

### Considerations

- A TLS listener requires a security policy. If you do not specify a security policy when you create the listener, we use the default security policy. The default security policy depends on how you created the TLS listener:
  - **Console** – The default security policy is `ELBSecurityPolicy-TLS13-1-2-Res-2021-06`.
  - **Other methods** (for example, the AWS CLI, AWS CloudFormation, and the AWS CDK) – The default security policy is `ELBSecurityPolicy-2016-08`.
- You can choose the security policy that is used for front-end connections, but not backend connections. The security policy for backend connections depends on the listener security policy:
  - If the TLS listener uses a TLS 1.3 security policy, backend connections use the `ELBSecurityPolicy-TLS13-1-0-2021-06` policy.
  - If the TLS listener does not use a TLS 1.3 security policy, backend connections use the `ELBSecurityPolicy-2016-08` policy.
- You can enable access logs for information about the TLS requests sent to your Network Load Balancer, analyze TLS traffic patterns, manage security policy upgrades, and troubleshoot issues. Enable access logging for your load balancer and examine the corresponding access log entries. For more information, see Access logs and Network Load Balancer Example Queries.
- You can restrict which security policies are available to users across your AWS accounts and AWS Organizations by using the Elastic Load Balancing condition keys in your IAM and service control policies (SCPs), respectively. For more information, see Service control policies (SCPs) in the *AWS Organizations User Guide*.
- Policies that support only TLS 1.3 support Forward Secrecy (FS). Policies that support TLS 1.3 and TLS 1.2 that have only ciphers of the form TLS_* and ECDHE_* also provide FS.
- Network Load Balancers support the Extended Master Secret (EMS) extension for TLS 1.2.

You can describe the protocols and ciphers using the describe-ssl-policies AWS CLI command, or refer to the tables below.

### Security policies

- TLS security policies
  - Protocols by policy

## TLS security policies

You can use the TLS security policies to meet compliance and security standards that require disabling certain TLS protocol versions, or to support legacy clients that require deprecated ciphers.

Policies that support only TLS 1.3 support Forward Secrecy (FS). Policies that support TLS 1.3 and TLS 1.2 that have only ciphers of the form TLS_* and ECDHE_* also provide FS.

**Contents**



### Protocols by policy

The following table describes the protocols that each TLS security policy supports.

| Security policies | TLS 1.3 | TLS 1.2 | TLS 1.1 | TLS 1.0 |
|---|---|---|---|---|
| ELBSecurityPolicy-TLS13-1-3-2021-06 | Yes | No | No | No |

| Security policies | TLS 1.3 | TLS 1.2 | TLS 1.1 | TLS 1.0 |
|---|---|---|---|---|
| ELBSecurityPolicy-TLS13-1-2-2021-06 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Res-2021-06 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-1-2021-06 | Yes | Yes | Yes | No |
| ELBSecurityPolicy-TLS13-1-0-2021-06 | Yes | Yes | Yes | Yes |
| ELBSecurityPolicy-TLS-1-2-Ext-2018-06 | No | Yes | No | No |
| ELBSecurityPolicy-TLS-1-2-2017-01 | No | Yes | No | No |
| ELBSecurityPolicy-TLS-1-1-2017-01 | No | Yes | Yes | No |
| ELBSecurityPolicy-2016-08 | No | Yes | Yes | Yes |
| ELBSecurityPolicy-2015-05 | No | Yes | Yes | Yes |

## Ciphers by policy

The following table describes the ciphers that each TLS security policy supports.

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-3-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256 |
| ELBSecurityPolicy-TLS13-1-2-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384 |
| ELBSecurityPolicy-TLS13-1-2-Res-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384 |
| ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| | • ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES256-GCM-SHA384<br>• AES256-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-1-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-0-2021-06 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• TLS_CHACHA20_POLY1305_SHA256<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS-1-2-Ext-2018-06 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS-1-2-2017-01 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES256-GCM-SHA384<br>• AES256-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS-1-1-2017-01 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2016-08 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-2015-05 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-ECDSA-AES256-SHA<br>• ECDHE-RSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

## Policies by cipher

The following table describes the TLS security policies that support each cipher.

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – TLS_AES_128_GCM_SHA256<br><br>**IANA** – TLS_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-3-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-2021-06 | 1301 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| | • ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06 | |
| **OpenSSL** – TLS_AES_256_GCM_SHA384<br><br>**IANA** – TLS_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-3-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06 | 1302 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – TLS_CHACHA20_POLY1305_SHA256<br><br>**IANA** – TLS_CHACHA20_POLY1305_SHA256 | • ELBSecurityPolicy-TLS13-1-3-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06 | 1303 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-GCM-SHA256<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c02b |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES128-GCM-SHA256<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c02f |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA256<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c023 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES128-SHA256<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c027 |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c009 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c013 |
| **OpenSSL** – ECDHE-ECDSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c02c |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Res-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c030 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c024 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS13-1-2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c028 |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c00a |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-RSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | c014 |
| **OpenSSL** – AES128-GCM-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 9c |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES128-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 3c |
| **OpenSSL** – AES128-SHA<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 2f |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES256-GCM-SHA384<br><br>**IANA** – TLS_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 9d |
| **OpenSSL** – AES256-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-2-2017-01<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 3d |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES256-SHA<br><br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06<br>• ELBSecurityPolicy-TLS13-1-1-2021-06<br>• ELBSecurityPolicy-TLS13-1-0-2021-06<br>• ELBSecurityPolicy-TLS-1-2-Ext-2018-06<br>• ELBSecurityPolicy-TLS-1-1-2017-01<br>• ELBSecurityPolicy-2016-08 | 35 |

## FIPS security policies

The Federal Information Processing Standard (FIPS) is a US and Canadian government standard that specifies the security requirements for cryptographic modules that protect sensitive information. To learn more, see Federal Information Processing Standard (FIPS) 140 on the *AWS Cloud Security Compliance* page.

All FIPS policies leverage the AWS-LC FIPS validated cryptographic module. To learn more, see the AWS-LC Cryptographic Module page on the *NIST Cryptographic Module Validation Program* site.

> **Important**
>
> Policies `ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04` and `ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04` are provided for legacy compatibility only. While they utilize FIPS cryptography using the FIPS140 module, they may not conform to the latest NIST guidance for TLS configuration.

**Contents**

- Policies by cipher

## Protocols by policy

The following table describes the protocols that each FIPS security policy supports.

| Security policies | TLS 1.3 | TLS 1.2 | TLS 1.1 | TLS 1.0 |
|---|---|---|---|---|
| ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04 | Yes | No | No | No |
| ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 | Yes | Yes | No | No |
| ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 | Yes | Yes | Yes | No |
| ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | Yes | Yes | Yes | Yes |

## Ciphers by policy

The following table describes the ciphers that each FIPS security policy supports.

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384 |
| ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384 |
| ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384 |
| ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384 |

| Security policy | Ciphers |
| --- | --- |
| | • ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |
| ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES256-GCM-SHA384<br>• AES256-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | • TLS_AES_128_GCM_SHA256<br>• TLS_AES_256_GCM_SHA384<br>• ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA<br>• AES128-GCM-SHA256<br>• AES128-SHA256<br>• AES128-SHA<br>• AES256-GCM-SHA384<br>• AES256-SHA256<br>• AES256-SHA |

## Policies by cipher

The following table describes the FIPS security policies that support each cipher.

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – TLS_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04 | 1301 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **IANA** – TLS_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | |
| **OpenSSL** – TLS_AES_256_GCM_SHA384<br><br>**IANA** – TLS_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 1302 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-GCM-SHA256<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c02b |
| **OpenSSL** – ECDHE-RSA-AES128-GCM-SHA256<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c02f |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA256<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c023 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA256<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c027 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c009 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c013 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c02c |
| **OpenSSL** – ECDHE-RSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c030 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c024 |
| **OpenSSL** – ECDHE-RSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c028 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c00a |
| **OpenSSL** – ECDHE-RSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | c014 |
| **OpenSSL** – AES128-GCM-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 9c |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES128-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 3c |
| **OpenSSL** – AES128-SHA<br><br>**IANA** – TLS_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 2f |
| **OpenSSL** – AES256-GCM-SHA384<br><br>**IANA** – TLS_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 9d |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – AES256-SHA256<br><br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA256 | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 3d |
| **OpenSSL** – AES256-SHA<br><br>**IANA** – TLS_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04<br>• ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 | 35 |

## FS supported security policies

FS (Forward Secrecy) supported security policies provide additional safeguards against the eavesdropping of encrypted data, through the use of a unique random session key. This prevents the decoding of captured data, even if the secret long-term key is compromised.

The policies in this section support FS, and "FS" is included in their names. However, these are not the only policies that support FS. Policies that support only TLS 1.3 support FS. Policies that support TLS 1.3 and TLS 1.2 that have only ciphers of the form TLS_* and ECDHE_* also provide FS.

**Contents**

## Protocols by policy

The following table describes the protocols that each FS supported security policy supports.

| Security policies | TLS 1.3 | TLS 1.2 | TLS 1.1 | TLS 1.0 |
|---|---|---|---|---|
| ELBSecurityPolicy-FS-1-2-Res-2020-10 | No | Yes | No | No |
| ELBSecurityPolicy-FS-1-2-Res-2019-08 | No | Yes | No | No |
| ELBSecurityPolicy-FS-1-2-2019-08 | No | Yes | No | No |
| ELBSecurityPolicy-FS-1-1-2019-08 | No | Yes | Yes | No |
| ELBSecurityPolicy-FS-2018-06 | No | Yes | Yes | Yes |

## Ciphers by policy

The following table describes the ciphers that each FS supported security policy supports.

| Security policy | Ciphers |
|---|---|
| ELBSecurityPolicy-FS-1-2-Res-2020-10 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384 |
| ELBSecurityPolicy-FS-1-2-Res-2019-08 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256 |

| Security policy | Ciphers |
| --- | --- |
| | • ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384 |
| ELBSecurityPolicy-FS-1-2-2019-08 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA |
| ELBSecurityPolicy-FS-1-1-2019-08 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA |

| Security policy | Ciphers |
| --- | --- |
| ELBSecurityPolicy-FS-2018-06 | • ECDHE-ECDSA-AES128-GCM-SHA256<br>• ECDHE-RSA-AES128-GCM-SHA256<br>• ECDHE-ECDSA-AES128-SHA256<br>• ECDHE-RSA-AES128-SHA256<br>• ECDHE-ECDSA-AES128-SHA<br>• ECDHE-RSA-AES128-SHA<br>• ECDHE-ECDSA-AES256-GCM-SHA384<br>• ECDHE-RSA-AES256-GCM-SHA384<br>• ECDHE-ECDSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA384<br>• ECDHE-RSA-AES256-SHA<br>• ECDHE-ECDSA-AES256-SHA |

## Policies by cipher

The following table describes the FS supported security policies that support each cipher.

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES128-GCM-SHA256<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-FS-1-2-Res-2020-10<br>• ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c02b |
| **OpenSSL** – ECDHE-RSA-AES128-GCM-SHA256 | • ELBSecurityPolicy-FS-1-2-Res-2020-10 | c02f |

| Cipher name | Security policies | Cipher suite |
|---|---|---|
| **IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 | • ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA256<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c023 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA256<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 | • ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c027 |
| **OpenSSL** – ECDHE-ECDSA-AES128-SHA<br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c009 |
| **OpenSSL** – ECDHE-RSA-AES128-SHA<br>**IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA | • ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c013 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-FS-1-2-Res-2020-10<br>• ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c02c |
| **OpenSSL** – ECDHE-RSA-AES256-GCM-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 | • ELBSecurityPolicy-FS-1-2-Res-2020-10<br>• ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c030 |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c024 |
| **OpenSSL** – ECDHE-RSA-AES256-SHA384<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 | • ELBSecurityPolicy-FS-1-2-Res-2019-08<br>• ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c028 |

| Cipher name | Security policies | Cipher suite |
| --- | --- | --- |
| **OpenSSL** – ECDHE-ECDSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c00a |
| **OpenSSL** – ECDHE-RSA-AES256-SHA<br><br>**IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA | • ELBSecurityPolicy-FS-1-2-2019-08<br>• ELBSecurityPolicy-FS-1-1-2019-08<br>• ELBSecurityPolicy-FS-2018-06 | c014 |

# Update a listener for your Network Load Balancer

You can update the listener protocol, listener port or the target group which receives traffic from the forwarding action. The default action, also known as the default rule, forwards requests to the selected 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.

When the target group for the default action of a TCP or TLS listener is updated, new connections are routed to the newly configured target group. However, this has no effect on any active connections that were created prior to this change. These active connections remain associated to the target in the original target group for up to one hour if traffic is being sent, or up to when the idle-timeout period elapses if no traffic is sent, whichever occurs first. The parameter `Connection termination on deregistration` is not applied when updating the listener, as it's applied when deregistering targets.

Console

**To update a listener**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.

3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. Choose **Edit**.
6. (Optional) Change the specified values for **Protocol** and **Port** as needed.
7. (Optional) Choose a different target group for **Default action**.
8. (Optional) Add, update, or remove tags as needed.
9. Choose **Save changes**.

AWS CLI

**To update the default action**

Use the following modify-listener command to change the target group for the default action.

```
aws elbv2 modify-listener \
    --listener-arn listener-arn \
    --default-actions Type=forward,TargetGroupArn=new-target-group-arn
```

**To add tags**

Use the add-tags command. The following example adds two tags.

```
aws elbv2 add-tags \
    --resource-arns listener-arn \
    --tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

**To remove tags**

Use the remove-tags command. The following example removes the tags with the specified keys.

```
aws elbv2 remove-tags \
    --resource-arns listener-arn \
    --tag-keys project department
```

CloudFormation

**To update the default action**

Update the AWS::ElasticLoadBalancingV2::Listener resource to include the new target group.

```
Resources:
  myTCPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TCP
      Port: 80
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref newTargetGroup
```

**To add tags**

Update the AWS::ElasticLoadBalancingV2::Listener resource to include the Tags property.

```
Resources:
  myTCPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TCP
      Port: 80
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
      Tags:
        - Key: 'project'
          Value: 'lima'
        - Key: 'department'
          Value: 'digital-media'
```

# Update the TCP idle timeout for your Network Load Balancer listener

For each TCP request made 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.

### Considerations

- The default idle timeout value for TCP flows is 350 seconds.
- The connection idle timeout for TLS listeners is 350 seconds and can't be modified.

Console

**To update the TCP idle timeout**

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 check box for the Network Load Balancer.
4. On the listeners tab, select the check box for the TCP listener and then choose **Actions**, **View listener details**.
5. On the listener details page, in the **Attributes** tab, select **Edit**. If the listener uses a protocol other than TCP, this tab is not present.
6. Enter a value for **TCP idle timeout** from 60-6000 seconds.
7. Choose **Save changes**.

AWS CLI

**To update the TCP idle timeout**

Use the modify-listener-attributes command with the `tcp.idle_timeout.seconds` attribute.

```
aws elbv2 modify-listener-attributes \
    --listener-arn listener-arn \
    --attributes Key=tcp.idle_timeout.seconds,Value=500
```

The following is example output.

```
{
    "Attributes": [
        {
            "Key": "tcp.idle_timeout.seconds",
            "Value": "500"
        }
    ]
```

```
}
```

CloudFormation

**To update the TCP idle timeout**

Update the AWS::ElasticLoadBalancingV2::Listener resource to include the `tcp.idle_timeout.seconds` listener attribute.

```
Resources:
  myTCPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TCP
      Port: 80
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
      ListenerAttributes:
        - Key: "tcp.idle_timeout.seconds"
          Value: "500"
```

# 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.

**Tasks**

- Replace the default certificate
- Add certificates to the certificate list
- Remove certificates from the certificate list
- Update the security policy
- Update the ALPN policy

# Replace the default certificate

You can replace the default certificate for your TLS listener as needed. For more information, see Default certificate.

Console

**To replace the default certificate**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Certificates** tab, choose **Change default**.
6. Within the **ACM and IAM certificates** table, select a new default certificate.
7. (Optional) By default, we select **Add previous default certificate to listener certificate list**. We recommend that you keep this option selected, unless you currently have no listener certificates for SNI and rely on TLS session resumption.
8. Choose **Save as default**.

AWS CLI

**To replace the default certificate**

Use the modify-listener command.

```
aws elbv2 modify-listener \
    --listener-arn listener-arn \
    --certificates CertificateArn=new-default-certificate-arn
```

CloudFormation

**To replace the default certificate**

Update the AWS::ElasticLoadBalancingV2::Listener resource with the new default certificate.

```
Resources:
```

```
myTLSListener:
  Type: 'AWS::ElasticLoadBalancingV2::Listener'
  Properties:
    LoadBalancerArn: !Ref myLoadBalancer
    Protocol: TLS
    Port: 443
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref myTargetGroup
    SslPolicy: "ELBSecurityPolicy-TLS13-1-2-2021-06"
    Certificates:
      - CertificateArn: "new-default-certificate-arn"
```

## 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 the default certificate to the certificate list 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.

Console

**To add certificates to the certificate list**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. Choose the **Certificates** tab.
6. To add the default certificate to the list, choose **Add default to list**.
7. To add nondefault certificates to the list, do the following:
   a. Choose **Add certificate**.
   b. To add certificates that are already managed by ACM or IAM, select the check boxes for the certificates and choose **Include as pending below**.
   c. To add a certificate that isn't managed by ACM or IAM, choose **Import certificate**, complete the form, and choose **Import**.

d. Choose **Add pending certificates**.

AWS CLI

**To add certificates to the certificate list**

Use the add-listener-certificates command.

```
aws elbv2 add-listener-certificates \
    --listener-arn listener-arn \
    --certificates \
        CertificateArn=certificate-arn-1 \
        CertificateArn=certificate-arn-2 \
        CertificateArn=certificate-arn-3
```

CloudFormation

**To add certificates to the certificate list**

Define a resource of type AWS::ElasticLoadBalancingV2::ListenerCertificate.

```
Resources:
  myCertificateList:
    Type: 'AWS::ElasticLoadBalancingV2::ListenerCertificate'
    Properties:
      ListenerArn: !Ref myTLSListener
      Certificates:
        - CertificateArn: "certificate-arn-1"
        - CertificateArn: "certificate-arn-2"
        - CertificateArn: "certificate-arn-3"

  myTLSListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TLSS
      Port: 443
      SslPolicy: "ELBSecurityPolicy-TLS13-1-2-2021-06"
      Certificates:
        - CertificateArn: "certificate-arn-1"
      DefaultActions:
        - Type: forward
```

```
            TargetGroupArn: !Ref myTargetGroup
```

## Remove certificates from the certificate list

You can remove certificates from the certificate list for a TLS listener using the following procedure. After you remove a certificate, the listener can no longer create connections using that certificate. To ensure that clients are not impacted, add a new certificate to the list and confirm that connections are working before you remove a certificate from the list.

To remove the default certificate for a TLS listener, see Replace the default certificate.

Console

**To remove certificates from the certificate list**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Certificates** tab, select the check boxes for the certificates and choose **Remove**.
6. When prompted for confirmation, enter **`confirm`** and choose **Remove**.

AWS CLI

**To remove certificates from the certificate list**

Use the remove-listener-certificates command.

```
aws elbv2 remove-listener-certificates \
    --listener-arn listener-arn \
    --certificates CertificateArn=certificate-arn
```

## 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 for your Network Load Balancer.

Updating the security policy can result in disruptions if the load balancer is handling a high volume of traffic. To decrease the possibility of disruptions when your load balancer is handling a high volume of traffic, create an additional load balancer to help handle the traffic or request an LCU reservation.

Console

**To update the security policy**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. Choose **Actions**, **Edit listener**.
6. In the **Secure listener settings** section, under **Security policy**, choose a new security policy.
7. Choose **Save changes**.

AWS CLI

**To update the security policy**

Use the modify-listener command.

```
aws elbv2 modify-listener \
    --listener-arn listener-arn \
    --ssl-policy ELBSecurityPolicy-TLS13-1-2-Res-2021-06
```

CloudFormation

**To update the security policy**

Update the AWS::ElasticLoadBalancingV2::Listener resource with the new security policy.

```
Resources:
  myTLSListener:
```

```
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TLS
      Port: 443
      SslPolicy: "ELBSecurityPolicy-TLS13-1-2-2021-06"
      Certificates:
        - CertificateArn: "default-certificate-arn"
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

## Update the ALPN policy

You can update the ALPN policy for your TLS listener as needed. For more information, see ALPN policies.

Console

**To update the ALPN policy**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose the name of the load balancer to open its detail page.
4. On the **Listeners** tab, choose the text in the **Protocol:Port** column to open the detail page for the listener.
5. Choose **Actions**, **Edit listener**.
6. In the **Secure listener settings** section, for **ALPN policy**, choose a policy to enable ALPN or choose **None** to disable ALPN.
7. Choose **Save changes**.

AWS CLI

**To update the ALPN policy**

Use the modify-listener command.

```
aws elbv2 modify-listener \
```

```
    --listener-arn listener-arn \
    --alpn-policy HTTP2Preferred
```

CloudFormation

**To update the ALPN policy**

Update the AWS::ElasticLoadBalancingV2::Listener resource to include the ALPN policy.

```
Resources:
  myTLSListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TLS
      Port: 443
      SslPolicy: "ELBSecurityPolicy-TLS13-1-2-Res-2021-06"
      AlpnPolicy:
        - HTTP2Preferred
      Certificates:
        - CertificateArn: "certificate-arn"
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

# Delete a listener for your Network Load Balancer

Before you delete a listener, consider the impact on your application:

- [TCP and TLS listeners] The load balancer immediately stops accepting new connections on the listener. Any TLS handshakes in progress might fail. Existing connections remain open until they naturally close or time out. In-flight requests on existing connections complete successfully.
- [UDP listeners] Any packets in transit might not reach their destination.

Console

**To delete a listener**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.

3. Select the check box for load balancer.
4. On the **Listeners** tab, select the check box for the listener, and then choose **Actions**, **Delete listener**.
5. When prompted for confirmation, enter **`confirm`** and choose **Delete**.

AWS CLI

**To delete a listener**

Use the delete-listener command.

```
aws elbv2 delete-listener \
    --listener-arn listener-arn
```

# 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.

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 Network Load Balancer target groups.

**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 |
| TCP | TCP | alb | HTTP \| HTTPS |
| 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 can't 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.

`alb`

The target is an Application Load Balancer.

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

- The subnets of the target group VPC
- 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.

All of the supported CIDR blocks enable you to register the following targets with a target group:

- AWS resources that are addressable by IP address and port (for example, databases).
- On-premises resources linked to AWS through AWS Direct Connect or a Site-to-Site VPN connection.

When client IP preservation is disabled for your target groups, the load balancer can support about 55,000 connections per minute for each combination of Network Load Balancer IP address and 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.

When launching a Network Load Balancer in a shared VPC (as a participant), you can only register targets in subnets that have been shared with you.

When the target type is `alb`, you can register a single Application Load Balancer as a target. For more information, see Use an Application Load Balancer as a target of a Network Load Balancer.

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

If you have microservices on instances that are registered with a Network Load Balancer, you can't 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.

## Request routing and IP addresses

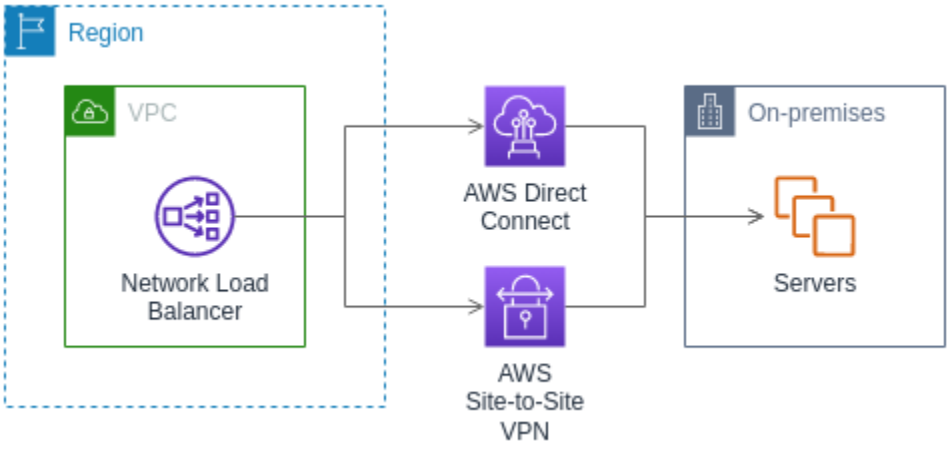
If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address that is specified in the primary network interface for the instance. The load balancer rewrites the destination IP address from the data packet before forwarding it to the target instance.

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 about allowing traffic to your instances, see Target security groups.

## On premises resources as targets

On premises resources linked through AWS Direct Connect or a Site-to-Site VPN connection can serve as a target, when the target type is `ip`.




When using on premises resources, the IP addresses of these targets must still come from one of the following CIDR blocks:

- 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)

For more information about AWS Direct Connect, see What is AWS Direct Connect?

For more information about AWS Site-to-Site VPN, see What is AWS Site-to-Site VPN?

## IP address type

When creating a new target group, you can select the IP address type of your target group. This controls the IP version used to communicate with targets and check their health status.

Target groups for your Network Load Balancers support the following IP address types:

**`ipv4`**

The load balancer communicates with targets using IPv4.

**`ipv6`**

The load balancer communicates with targets using IPv6.

**Considerations**

- The load balancer communicates with targets based on the IP address type of the target group. The targets of an IPv4 target group must accept IPv4 traffic from the load balancer and the targets of an IPv6 target group must accept IPv6 traffic from the load balancer.
- You can't use an IPv6 target group with an `ipv4` load balancer.
- You can't use an IPv4 target group with a UDP listener for a `dualstack` load balancer.
- You can't register an Application Load Balancer with an IPv6 target group.

# 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 and the target passes the first initial health check, irrespective of the configured threshold.

If demand on your application decreases, or if 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 and considerations**

- You can't 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.

- When registering targets by instance ID for a IPv6 target group, the targets must have an assigned primary IPv6 address. To learn more, see IPv6 addresses in the *Amazon EC2 User Guide*
- When registering targets by instance ID, instances must be in the same VPC as the Network Load Balancer. You can't register instances by instance ID if they are in an 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.
- The load balancer routes traffic to targets only in Availability Zones that are enabled. Targets in zones that are not enabled are unused.
- 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

You can configure a target group by editing its attributes. For more information, see Edit target group attributes.

The following target group attributes are supported. You can modify these attributes only if the target group type is `instance` or `ip`. If the target group type is `alb`, these attributes always use their default values.

`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`. For new UDP/TCP_UDP target groups the default is `true`. Otherwise, the default is `false`.

`load_balancing.cross_zone.enabled`

Indicates whether cross zone load balancing is enabled. The value is `true`, `false` or `use_load_balancer_configuration`. The default is `use_load_balancer_configuration`.

`preserve_client_ip.enabled`

Indicates whether client IP preservation is enabled. The value is `true` or `false`. The default is disabled if the target group type is IP address and the target group protocol is TCP or TLS. Otherwise, the default is enabled. Client IP preservation can't be disabled for UDP and TCP_UDP target groups.

`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. The value is `true` or `false`. The default is `false`.

`stickiness.type`

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

`target_group_health.dns_failover.minimum_healthy_targets.count`

The minimum number of targets that must be healthy. If the number of healthy targets is below this value, mark the zone as unhealthy in DNS, so that traffic is routed only to healthy zones. The possible values are `off` or an integer from 1 to the maximum number of targets. When `off`, DNS fail away is disabled, meaning that even if all targets in the target group are unhealthy, the zone is not removed from DNS. The default is 1.

`target_group_health.dns_failover.minimum_healthy_targets.percentage`

The minimum percentage of targets that must be healthy. If the percentage of healthy targets is below this value, mark the zone as unhealthy in DNS, so that traffic is routed only to healthy zones. The possible values are `off` or an integer from 1 to 100. When `off`, DNS fail away is disabled, meaning that even if all targets in the target group are unhealthy, the zone is not removed from DNS. The default is `off`.

`target_group_health.unhealthy_state_routing.minimum_healthy_targets.count`

The minimum number of targets that must be healthy. If the number of healthy targets is below this value, send traffic to all targets, including unhealthy targets. The possible values are 1 to the maximum number of targets. The default is 1.

`target_group_health.unhealthy_state_routing.minimum_healthy_targets.percentage`

The minimum percentage of targets that must be healthy. If the percentage of healthy targets is below this value, send traffic to all targets, including unhealthy targets. The possible values are `off` or an integer from 1 to 100. The default is `off`.

`target_health_state.unhealthy.connection_termination.enabled`

Indicates whether the load balancer terminates connections to unhealthy targets. The value is `true` or `false`. The default is `true`.

`target_health_state.unhealthy.draining_interval_seconds`

The amount of time for Elastic Load Balancing to wait before changing the state of an unhealthy target from `unhealthy.draining` to `unhealthy`. The range is 0-360000 seconds. The default value is 0 seconds.

**Note:** This attribute can only be configured when `target_health_state.unhealthy.connection_termination.enabled` is `false`.

# Target group health

By default, a target group is considered healthy as long as it has at least one healthy target. If you have a large fleet, having only one healthy target serving traffic is not sufficient. Instead, you can specify a minimum count or percentage of targets that must be healthy, and what actions the load balancer takes when the healthy targets fall below the specified threshold. This improves the availability of your application.

**Contents**

## Unhealthy state actions

You can configure healthy thresholds for the following actions:

- **DNS failover** – When the healthy targets in a zone fall below the threshold, we mark the IP addresses of the load balancer node for the zone as unhealthy in DNS. Therefore, when clients resolve the load balancer DNS name, the traffic is routed only to healthy zones.
- **Routing failover** – When the healthy targets in a zone fall below the threshold, the load balancer sends traffic to all targets that are available to the load balancer node, including unhealthy targets. This increases the chances that a client connection succeeds, especially when targets temporarily fail to pass health checks, and reduces the risk of overloading the healthy targets.

## Requirements and considerations

- If you specify both types of thresholds for an action (count and percentage), the load balancer takes the action when either threshold is breached.
- If you specify thresholds for both actions, the threshold for DNS failover must be greater than or equal to the threshold for routing failover, so that DNS failover occurs either with or before routing failover.
- If you specify the threshold as a percentage, we calculate the value dynamically, based on the total number of targets that are registered with the target groups.
- The total number of targets is based on whether cross-zone load balancing is off or on. If cross-zone load balancing is off, each node sends traffic only to the targets in its own zone, which means that the thresholds apply to the number of targets in each enabled zone separately. If cross-zone load balancing is on, each node sends traffic to all targets in all enabled zones, which means that the specified thresholds apply to the total number targets in all enabled zones. For more information, see Cross-zone load balancing.
- When DNS failover occurs, it impacts all target groups associated with the load balancer. Ensure that you have enough capacity in your remaining zones to handle this additional traffic, especially if cross-zone load balancing is off.
- With DNS failover, we remove the IP addresses of the unhealthy zones from the DNS hostname for the load balancer. However, the local client DNS cache might contain these IP addresses until the time-to-live (TTL) in the DNS record expires (60 seconds).

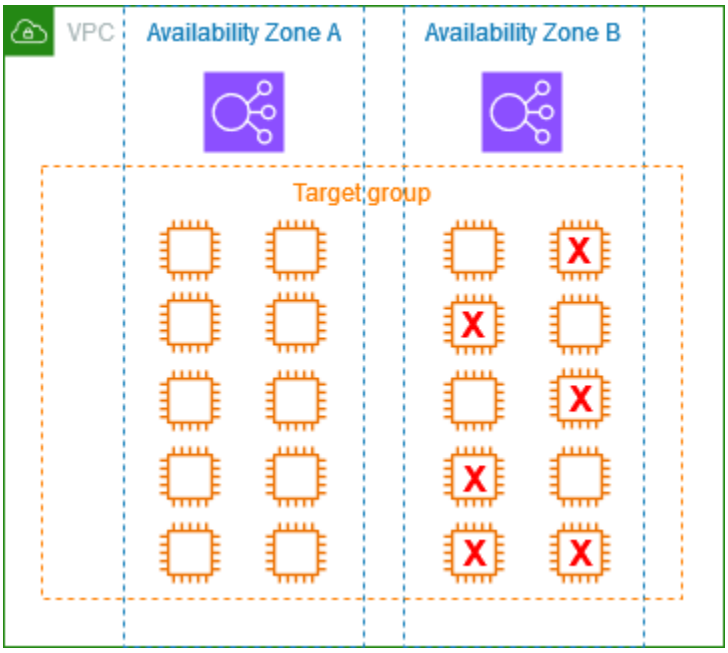
- With DNS failover, if there are multiple target groups attached to a Network Load Balancer and one target group is unhealthy in a zone, DNS failover occurs, even if another target group is healthy in that zone.
- With DNS failover, if all load balancer zones are considered unhealthy, the load balancer sends traffic to all zones, including the unhealthy zones.
- There are factors other than whether there are enough healthy targets that might lead to DNS failover, such as the health of the zone.

## Example

The following example demonstrates how target group health settings are applied.

### Scenario

- A load balancer that supports two Availability Zones, A and B
- Each Availability Zone contains 10 registered targets
- The target group has the following target group health settings:
  - DNS failover - 50%
  - Routing failover - 50%
- Six targets fail in Availability Zone B



**If cross-zone load balancing is off**

- The load balancer node in each Availability Zone can send traffic only to the 10 targets in its Availability Zone.
- There are 10 healthy targets in Availability Zone A, which meets the required percentage of healthy targets. The load balancer continues to distribute traffic between the 10 healthy targets.
- There are only 4 healthy targets in Availability Zone B, which is 40% of the targets for the load balancer node in Availability Zone B. Because this is less than the required percentage of healthy targets, the load balancer takes the following actions:
  - DNS failover - Availability Zone B is marked as unhealthy in DNS. Because clients can't resolve the load balancer name to the load balancer node in Availability Zone B, and Availability Zone A is healthy, clients send new connections to Availability Zone A.
  - Routing failover - When new connections are sent explicitly to Availability Zone B, the load balancer distributes traffic to all targets in Availability Zone B, including the unhealthy targets. This prevents outages among the remaining healthy targets.

**If cross-zone load balancing is on**

- Each load balancer node can send traffic to all 20 registered targets across both Availability Zones.
- There are 10 healthy targets in Availability Zone A and 4 healthy targets in Availability Zone B, for a total of 14 healthy targets. This is 70% of the targets for the load balancer nodes in both Availability Zones, which meets the required percentage of healthy targets.
- The load balancer distributes traffic between the 14 healthy targets in both Availability Zones.

## Using Route 53 DNS failover for your load balancer

If you use Route 53 to route DNS queries to your load balancer, you can also configure DNS failover for your load balancer using Route 53. In a failover configuration, Route 53 checks the health of the target group targets for the load balancer to determine whether they are available. If there are no healthy targets registered with the load balancer, or if the load balancer itself is unhealthy, Route 53 routes traffic to another available resource, such as a healthy load balancer or a static website in Amazon S3.

For example, suppose that you have a web application for `www.example.com`, and you want redundant instances running behind two load balancers residing in different Regions. You want

the traffic to be primarily routed to the load balancer in one Region, and you want to use the load balancer in the other Region as a backup during failures. If you configure DNS failover, you can specify your primary and secondary (backup) load balancers. Route 53 directs traffic to the primary load balancer if it is available, or to the secondary load balancer otherwise.

**How evaluate target health works**

- If evaluate target health is set to `Yes` on an alias record for a Network Load Balancer, Route 53 evaluates the health of the resource specified by the `alias target` value. Route 53 uses the target group health checks.
- If all target groups attached to a Network Load Balancer are healthy, Route 53 marks the alias record as healthy. If you configured a threshold for a target group and it meets its threshold, it passes health checks. Otherwise, if a target group contains at least one healthy target, it passes health checks. If health checks pass, Route 53 returns records according to your routing policy. If a failover routing policy is used, Route 53 returns the primary record.
- If all target groups attached to a Network Load Balancer are unhealthy, the alias record fails the Route 53 health check (fail-open). If using evaluate target health, this causes the failover routing policy to redirect traffic to the secondary resource.
- If all of the target groups in a Network Load Balancer are empty (no targets), Route 53 considers the record unhealthy (fail-open). If using evaluate target health, this causes the failover routing policy to redirect traffic to the secondary resource.

For more information, see Using load balancer target group health thresholds to improve availability in the AWS Blog and Configuring DNS failover in the *Amazon Route 53 Developer Guide*.

# 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.

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. You can specify the same target group in multiple listeners, but these listeners must belong to the same Network Load Balancer. To use a target group with a load balancer, you must verify that the target group is not in use by a listener for any other load balancer.

You can add or remove targets from your target group at any time. For more information, see Register targets for your Network Load Balancer. You can also modify the health check settings for your target group. For more information, see Update the health check settings of a Network Load Balancer target group.

**Requirements**

- After you create a target group, you can't change its target type or its IP address type.
- All targets in a target group must have the same IP address type as the target group: IPv4 or IPv6.
- You must use an IPv6 target group with a dualstack load balancer.
- You can't use an IPv4 target group with a UDP listener for a `dualstack` load balancer.

Console

**To create a target group**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Target Groups**.
3. Choose **Create target group**.
4. For the **Basic configuration** pane, do the following:
   a. For **Choose a target type**, select **Instances** to register targets by instance ID, **IP addresses** to register targets by IP address, or **Application Load Balancer** to register an Application Load Balancer as a target.
   b. For **Target group name**, enter 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.
   c. 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**.
      - If the target type is **Application Load Balancer**, the protocol must be TCP.
   d. For **Port**, modify the default value as needed.

If the target type is **Application Load Balancer**, the port must match the listener port of the Application Load Balancer.

e. For **IP address type**, choose **IPv4** or **IPv6**. This option is available only if the target type is **Instances** or **IP addresses**.

f. For **VPC**, select the virtual private cloud (VPC) with the targets to register.

5. For the **Health checks** pane, modify the default settings as needed. For **Advanced health check settings**, choose the health check port, count, timeout, interval, and specify success codes. If health checks consecutively exceed the **Unhealthy threshold** count, the load balancer takes the target out of service. If health checks consecutively exceed the **Healthy threshold** count, the load balancer puts the target back in service. For more information, see ???.
6. (Optional) To add a tag, expand **Tags**, choose **Add tag**, and enter a tag key and a tag value.
7. Choose **Next**.
8. (Optional) Register targets. The target type of the target group determines the information that you provide. If you aren't ready to register targets now, you can register them later.
   - **Instances** – Select the EC2 instances, enter the ports, and choose **Include as pending below**.
   - **IP addresses** – Choose the VPC that contains the IP addresses or **Other private IP addresses**, enter the IP addresses and ports, and choose **Include as pending below**.
   - **Application Load Balancer** – Select the Application Load Balancer. For more information, see Use Application Load Balancers as targets.
9. Choose **Create target group**.

AWS CLI

**To create a target group**

Use the create-target-group command. The following example creates a target group with the TCP protocol, targets registered by IP address, one tag, and default health check settings.

```
aws elbv2 create-target-group \
    --name my-target-group \
    --protocol TCP \
    --port 80 \
    --target-type ip \
```

```
    --vpc-id vpc-1234567890abcdef0 \
    --tags Key=department,Value=123
```

**To register targets**

Use the register-targets command to register targets with the target group. For examples, see the section called “Register targets”.

CloudFormation

**To create a target group**

Define a resource of type AWS::ElasticLoadBalancingV2::TargetGroup. The following example creates a target group with the TCP protocol, targets registered by IP address, one tag, default health check settings, and two registered targets.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      Tags:
        - Key: 'department'
          Value: '123'
      Targets:
        - Id: 10.0.50.10
          Port: 80
        - Id: 10.0.50.20
          Port: 80
```

# Update the target group health settings for your Network Load Balancer

By default, Network Load Balancers monitor the health of targets and route requests to healthy targets. However, if the load balancer doesn't have enough healthy targets, it automatically sends traffic to all registered targets (fail open). You can modify the target group health settings for your

target group to define the thresholds for DNS failover and routing failover. For more information, see the section called "Target group health".

Console

**To update the target group health settings**

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 **Attributes** tab, choose **Edit**.
5. Expand **Target group health requirements**.
6. For **Configuration type**, we recommend that you choose **Unified configuration**, which sets the same threshold for both DNS failover and routing failover.
7. For **Healthy state requirements**, do one of the following:
   - Choose **Minimum healthy target count**, and then enter a number from 1 to the maximum number of targets for your target group.
   - Choose **Minimum healthy target percentage**, and then enter a number from 1 to 100.
8. The informational text indicates whether cross-zone load balancing is enabled for the target group. If cross-zone load balancing is disabled, you can enable it to ensure that you have enough capacity. Under **Target selection configuration**, update **Cross-zone load balancing**.

   The following text indicates that cross-zone load balancing is disabled:

   ```
   Healthy state requirements apply to each zone independently.
   ```

   The following text indicates that cross-zone load balancing is enabled:

   ```
   Healthy state requirements apply to the total targets across all applicable
    zones.
   ```
9. Choose **Save changes**.

AWS CLI

**To update the target group health settings**

Use the modify-target-group-attributes command. The following example sets the healthy threshold for both unhealthy state actions to 50%.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes \

 "Key=target_group_health.dns_failover.minimum_healthy_targets.percentage,Value=50"
 \

 "Key=target_group_health.unhealthy_state_routing.minimum_healthy_targets.percentage,Value=5
```

CloudFormation

**To modify target group health settings**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource. The following example sets the healthy threshold for both unhealthy state actions to 50%.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "target_group_health.dns_failover.minimum_healthy_targets.percentage"
          Value: "50"
        - Key:
 "target_group_health.unhealthy_state_routing.minimum_healthy_targets.percentage"
          Value: "50"
```

# Health checks for Network Load Balancer 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 and the targets pass the initial health checks. 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.

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, and target groups with stickiness turned on. For more information, see Sticky sessions.

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

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. Network Load Balancers will also fail open when you have an empty target group. The effect of the fail open is to allow traffic to all targets in all enabled Availability Zones, regardless of their health status.

If a target group is configured with HTTPS health checks, its registered targets fail health checks if they support only TLS 1.3. These targets must support an earlier version of TLS, such as TLS 1.2.

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 connections for this test sent from your load balancer to the targets that are registered with your TLS listener. You can identify these TCP connections 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.

For more information, see the section called “Target group health”.

**Contents**



## 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 | Default |
| --- | --- | --- |
| **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. If the target type is `alb`, the supported health check protocols are HTTP and HTTPS. | TCP |
| **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. | Port on which each target receives traffic from the load balancer. |

| Setting | Description | Default |
| --- | --- | --- |
| **HealthCheckPath** | [HTTP/HTTPS health checks] The health check 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 check. The range is 2–120 seconds. The default values are 6 seconds for HTTP and 10 seconds for TCP and HTTPS health checks. | 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. The range is 5–300 seconds. The default is 30 seconds.<br><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. | 30 seconds |
| **HealthyThresholdCount** | The number of consecutive successful health checks required before considering an unhealthy target healthy. The range is 2–10. The default is 5. | 5 |

| Setting | Description | Default |
| --- | --- | --- |
| **UnhealthyThresholdCount** | The number of consecutive failed health checks required before considering a target unhealthy. The range is 2–10. The default is 2. | 2 |
| **Matcher** | [HTTP/HTTPS health checks] The HTTP codes to use when checking for a successful response from a target. The range is 200 to 599. The default is 200-399. | 200-399 |

## 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 |
| `unhealthy` | The target did not respond to a health check, failed the health check, or the target is in stopped state.<br><br>Related reason code: `Target.FailedHealthChecks` |

| Value | Description |
| --- | --- |
| `draining` | The target is deregistering and connection draining is in process.<br><br>Related reason code: `Target.DeregistrationInProgress` |
| `unhealthy.draining` | The target did not respond to health checks or failed the health checks and enters a grace period. The target supports existing connections and will not accept any new connections during this grace period.<br><br>Related reason code: `Target.FailedHealthChecks` |
| `unavailable` | Target health is unavailable.<br><br>Related reason code: `Elb.InternalError` |
| `unused` | The target is not registered with a target group, the target group is not used in a listener rule, or the target is in an Availability Zone that is not enabled.<br><br>Related reason codes: `Target.NotRegistered` \| `Target.NotInUse` \| `Target.InvalidState` \| `Target.IpUnusable` |

## 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 |

| Reason code | Description |
| --- | --- |
| `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>Target is in the terminated state<br>Target is in the terminated or stopped state<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<br>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 Network Load Balancer targets

You can check the health status of the targets registered with your target groups. For help with health check failures, see Troubleshooting: A registered target is not in service.

Console

**To check the health of your targets**

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. The **Details** tab displays the total number of targets, plus the number of targets for each health status.
5. On the **Targets** tab, the **Health status** column indicates the status of each target.
6. If the status of a target is any value other than `Healthy`, the **Health status details** column contains more information.

**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.

AWS CLI

**To check the health of your targets**

Use the describe-target-health command. This example filters the output to include only targets that are not healthy. For targets that are not healthy, the output includes a reason code.

```
aws elbv2 describe-target-health \
    --target-group-arn target-group-arn \
    --query "TargetHealthDescriptions[?TargetHealth.State!='healthy'].
[Target.Id,TargetHealth.State,TargetHealth.Reason]" \
    --output table
```

The following is example output.

```
-------------------------------------------------
|            DescribeTargetHealth              |
+--------------+---------+--------------------+
|  172.31.0.57 |  unused |  Target.NotInUse   |
|  172.31.0.50 |  unused |  Target.NotInUse   |
```

```
+--------------+---------+-------------------+
```

### Target states and reason codes

The following list shows the possible reason codes for each target state.

**Target state is healthy**

A reason code is not provided.

**Target state is initial**

- `Elb.RegistrationInProgress` - The target is in the process of being registered with the load balancer.
- `Elb.InitialHealthChecking` - The load balancer is still sending the target the minimum number of health checks required to determine its health status.

**Target state is unhealthy**

- `Target.FailedHealthChecks` - The load balancer received an error while establishing a connection to the target or the target response was malformed.

**Target state is unused**

- `Target.NotRegistered` - The target is not registered with the target group.
- `Target.NotInUse` - The target group is not used by any load balancer or the target is in an Availability Zone that is not enabled for its load balancer.
- `Target.InvalidState` - The target is in the stopped or terminated state.
- `Target.IpUnusable` - The target IP address is reserved for use by a load balancer.

**Target state is draining**

- `Target.DeregistrationInProgress` - The target is in the process of being deregistered and the deregistration delay period has not expired.

**Target state is unavailable**

- `Elb.InternalError` - Target health is unavailable due to an internal error.

## Update the health check settings of a Network Load Balancer target group

You can update the health check settings for your target group at any time. For the list of health check settings, see the section called “Health check settings”.

Console

**To update the health check settings**

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 **Health checks** tab, choose **Edit**.
5. On the **Edit health check settings** page, modify the settings as needed.
6. Choose **Save changes**.

AWS CLI

**To update the health check settings**

Use the modify-target-group command. The following example updates the **HealthyThresholdCount** and **HealthCheckTimeoutSeconds** settings.

```
aws elbv2 modify-target-group \
    --target-group-arn target-group-arn \
    --healthy-threshold-count 3 \
    --health-check-timeout-seconds 20
```

CloudFormation

**To update the health check settings**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the updated health check settings. The following example updates the **HealthyThresholdCount** and **HealthCheckTimeoutSeconds** settings.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: instance
```

```
      VpcId: !Ref myVPC
      HealthyThresholdCount: 3
      HealthCheckTimeoutSeconds: 20
```

# Edit target group attributes for your Network Load Balancer

After you create a target group for your Network Load Balancer, you can edit its target group attributes.

**Target group attributes**

- Client IP preservation
- Deregistration delay
- Proxy protocol
- Sticky sessions
- Cross-zone load balancing for target groups
- Connection termination for unhealthy targets
- Unhealthy draining interval

## Client IP preservation

Network Load Balancers can preserve the source IP addresses of clients when routing requests to backend targets. When you disable client IP preservation, the source IP address is the private IP address of the Network Load Balancer.

By default, client IP preservation is enabled (and can't be disabled) for instance and IP type target groups with UDP and TCP_UDP protocols. However, you can enable or disable client IP preservation for TCP and TLS target groups using the `preserve_client_ip.enabled` target group attribute.

**Default settings**

- Instance type target groups: Enabled
- IP type target groups (UDP, TCP_UDP): Enabled
- IP type target groups (TCP, TLS): Disabled

**When client IP preservation is enabled**

The following table describes the IP addresses that targets receive when client IP preservation is enabled.

| Targets | IPv4 client requests | IPv6 client requests |
|---|---|---|
| Instance type (IPv4) | Client IPv4 address | Load balancer IPv4 address |
| IP type (IPv4) | Client IPv4 address | Load balancer IPv4 address |
| IP type (IPv6) | Load balancer IPv6 address | Client IPv6 address |

### When client IP preservation is disabled

The following table describes the IP addresses that targets receive when client IP preservation is disabled.

| Targets | IPv4 client requests | IPv6 client requests |
|---|---|---|
| Instance type (IPv4) | Load balancer IPv4 address | Load balancer IPv4 address |
| IP type (IPv4) | Load balancer IPv4 address | Load balancer IPv4 address |
| IP type (IPv6) | Load balancer IPv6 address | Load balancer IPv6 address |

### Requirements and considerations

- Client IP preservation changes take effect only for new TCP connections.
- When client IP preservation is enabled, the traffic must flow directly from the Network Load Balancer to the target. The target must be located in the same VPC as the load balancer or in a peered VPC in the same Region.
- Client IP preservation is not supported when targets are reached through a transit gateway.
- Client IP preservation is not supported when using a Gateway Load Balancer endpoint to inspect traffic between the Network Load Balancer and the target (instance or IP address), even if the target is in the same VPC as the Network Load Balancer.
- The following instance types do not support client IP preservation: C1, CC1, CC2, CG1, CG2, CR1, G1, G2, HI1, HS1, M1, M2, M3, and T1. We recommend that you register these instance types as IP addresses with client IP preservation disabled.

- Client IP preservation has no effect on inbound traffic from AWS PrivateLink. The source IP address of the AWS PrivateLink traffic is always the private IP address of the Network Load Balancer.
- Client IP preservation is not supported when a target group contains AWS PrivateLink network interfaces, or the network interface of another Network Load Balancer. This causes a loss of communication to those targets.
- Client IP preservation has no effect on traffic converted from IPv6 to IPv4. The source IP address of this type of traffic is always the private IP address of the Network Load Balancer.
- When you specify targets by Application Load Balancer type, the client IP of all incoming traffic is preserved by the Network Load Balancer and is sent to the Application Load Balancer. The Application Load Balancer then appends the client IP to the `X-Forwarded-For` request header before sending it to the target.
- NAT loopback, also known as hairpinning, is not supported when client IP preservation is enabled. This occurs when using internal Network Load Balancers, and the target registered behind a Network Load Balancer creates connections to the same Network Load Balancer. The connection can be routed to the target which is attempting to create the connection, leading to connection errors. We recommend not connecting to a Network Load Balancer from targets behind same Network Load Balancer, alternatively you can also prevent this type of connection error by disabling client IP preservation. If you need the client IP address, you can use retrieve it using Proxy Protocol v2. For more information, see Proxy protocol.
- When client IP preservation is disabled, a Network Load Balancer 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, resulting in failures to establish new connections. For more information, see Port allocation errors for backend flows.

Console

**To modify client IP preservation**

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 **Attributes** tab, choose **Edit** and find the **Traffic configuration** pane.

5. To enable client IP preservation, turn on **Preserve client IP addresses**. To disable client IP preservation, turn off **Preserve client IP addresses**.
6. Choose **Save changes**.

AWS CLI

**To enable client IP preservation**

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

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes "Key=preserve_client_ip.enabled,Value=true"
```

CloudFormation

**To enable client IP preservation**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `preserve_client_ip.enabled` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "preserve_client_ip.enabled"
          Value: "true"
```

## Deregistration delay

When a target is deregistered, 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`, during which the target will stop receiving new connections. However, the target may still receive connections due to configuration propagation delay. 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.

Console

**To modify the deregistration delay attributes**

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 **Attributes** tab, 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 **Terminate connections on deregistration**.
6. Choose **Save changes**.

AWS CLI

**To modify the deregistration delay attributes**

Use the modify-target-group-attributes command with the `deregistration_delay.timeout_seconds` and `deregistration_delay.connection_termination.enabled` attributes.

```
aws elbv2 modify-target-group-attributes \
```

```
    --target-group-arn target-group-arn \
    --attributes \
      "Key=deregistration_delay.timeout_seconds,Value=60" \
      "Key=deregistration_delay.connection_termination.enabled,Value=true"
```

CloudFormation

**To modify the deregistration delay attributes**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `deregistration_delay.timeout_seconds` and `deregistration_delay.connection_termination.enabled` attributes.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "deregistration_delay.timeout_seconds"
          Value: "60"
        - Key: "deregistration_delay.connection_termination.enabled"
          Value: "true"
```

## 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.

With TCP listeners, the load balancer prepends a proxy protocol header to the TCP data. It does not discard or overwrite any existing data, including any incoming 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 the load balancer.

TLS listeners do not support incoming connections with proxy protocol headers sent by the client or any other proxies.

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: By default, client IP preservation is disabled, and the source IP addresses provided to your applications are the private IP addresses of the load balancer nodes. To preserve the client's IP address, ensure that the target is located within the same VPC or a peered VPC and enable client IP preservation. If you need the IP address of the client and these conditions are not met, enable the proxy protocol and get the client IP address from the proxy protocol header.
- UDP and TCP_UDP: The source IP addresses are the IP addresses of the clients, as client IP preservation is enabled by default for these protocols and cannot be disabled. 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.

Targets can fail health checks if they can't parse the proxy protocol header. For example, they might return the following error: HTTP 400: Bad request.

## 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) |

| Field | Length (in octets) | Description |
| --- | --- | --- |
| 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.

Console

**To enable proxy protocol version 2**

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 **Attributes** tab, choose **Edit**.
5. On the **Edit attributes** page, select **Proxy protocol v2**.
6. Choose **Save changes**.

AWS CLI

**To enable proxy protocol version 2**

Use the modify-target-group-attributes command with the `proxy_protocol_v2.enabled` attribute.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
```

```
    --attributes "Key=proxy_protocol_v2.enabled,Value=true"
```

CloudFormation

**To enable proxy protocol version 2**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `proxy_protocol_v2.enabled` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "proxy_protocol_v2.enabled"
          Value: "true"
```

## 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.
- When the stickiness attribute is turned on for a target group, passive health checks are not supported. For more information, see Health checks for your target groups.
- Sticky sessions are not supported for TLS listeners.

Console

**To enable sticky sessions**

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 **Attributes** tab, choose **Edit**.
5. Under **Target selection configuration**, turn on **Stickiness**.
6. Choose **Save changes**.

AWS CLI

**To enable sticky sessions**

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

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes "Key=stickiness.enabled,Value=true"
```

CloudFormation

**To enable sticky sessions**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `stickiness.enabled` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "stickiness.enabled"
          Value: "true"
```

# Cross-zone load balancing for target groups

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is on, each load balancer node distributes traffic across the registered targets in all registered Availability Zones. When cross-zone load balancing is off, each load balancer node distributes traffic across only the registered targets in its Availability Zone. This could be used if zonal failure domains are preferred over regional, ensuring that a healthy zone isn't impacted by an unhealthy zone, or for overall latency improvements.

With Network Load Balancers, cross-zone load balancing is disabled by default at the load balancer level, bit you can enable it at any time. For target groups, the default is to use the load balancer setting, but you can override the default by explicitly enabling or disabling cross-zone load balancing at the target group level.

**Considerations**

- When enabling cross-zone load balancing for a Network Load Balancer, EC2 data transfer charges apply. For more information, see Understanding data transfer charges in the *AWS Data Exports User Guide*
- The target group setting determines the load balancing behavior for the target group. For example, if cross-zone load balancing is enabled at the load balancer level and disabled at the target group level, traffic sent to the target group is not routed across Availability Zones.
- When cross-zone load balancing is disabled, ensure that you have enough target capacity in each of the load balancer Availability Zones, so that each zone can serve its associated workload.
- When cross-zone load balancing is disabled, ensure that all target groups participate in the same Availability Zones. An empty Availability Zone is considered unhealthy.
- You can enable or disable cross-zone load balancing at the target group level if the target group type is `instance` or `ip`. If the target group type is `alb`, the target group always inherits the cross-zone load balancing setting from the load balancer.

For more information about enabling cross-zone load balancing at the load balancer level, see the section called “Cross-zone load balancing”.

Console

**To enable cross-zone load balancing for a target group**

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

2. On the navigation pane, under **Load Balancing**, select **Target Groups**.
3. Select the name of the target group to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. On the **Edit target group attributes** page, select **On** for **Cross-zone load balancing**.
6. Choose **Save changes**.

AWS CLI

**To enable cross-zone load balancing for a target group**

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

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes "Key=load_balancing.cross_zone.enabled,Value=true"
```

CloudFormation

**To enable cross-zone load balancing for a target group**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `load_balancing.cross_zone.enabled` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "load_balancing.cross_zone.enabled"
          Value: "true"
```

# Connection termination for unhealthy targets

Connection termination is enabled by default. When the target of a Network Load Balancer fails the configured health checks and is deemed unhealthy, the load balancer terminates established connections and stops routing new connections to the target. With connection termination disabled the target is still considered unhealthy and won't receive new connections, but established connections are kept active, allowing them to gracefully close.

Connection termination for unhealthy targets is configured at the target group level.

Console

**To modify the connection termination attribute**

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 **Attributes** tab, choose **Edit**.
5. Under **Target unhealthy state management**, choose whether **Terminate connections when targets become unhealthy** is enabled or disabled.
6. Choose **Save changes**.

AWS CLI

**To disable the connection termination attribute**

Use the modify-target-group-attributes command with the `target_health_state.unhealthy.connection_termination.enabled` attribute.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes
 "Key=target_health_state.unhealthy.connection_termination.enabled,Value=false"
```

CloudFormation

**To disable the connection termination attribute**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `target_health_state.unhealthy.connection_termination.enabled` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "target_health_state.unhealthy.connection_termination.enabled"
          Value: "false"
```

## Unhealthy draining interval

Targets in the `unhealthy.draining` state are considered unhealthy, do not receive new connections, but retain established connections for the configured interval. The unhealthy connection interval determines the amount of time the target remains in the `unhealthy.draining` state before its state becomes `unhealthy`. If the target passes health checks during the unhealthy connection interval, its state becomes `healthy` again. If a deregistration is triggered, the targets state becomes `draining` and the deregistration delay timeout begins.

### Requirement

Connection termination must be disabled before enabling unhealthy draining interval.

Console

#### To modify the unhealthy draining interval

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 **Attributes** tab, choose **Edit**.

5. Under **Target unhealthy state management**, make sure **Terminate connections when targets become unhealthy** is turned off.
6. Enter a value for **Unhealthy draining interval**.
7. Choose **Save changes**.

AWS CLI

**To modify the unhealthy draining interval**

Use the modify-target-group-attributes command with the `target_health_state.unhealthy.draining_interval_seconds` attribute.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes
 "Key=target_health_state.unhealthy.draining_interval_seconds,Value=60"
```

CloudFormation

**To modify the unhealthy draining interval**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `target_health_state.unhealthy.draining_interval_seconds` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "target_health_state.unhealthy.draining_interval_seconds"
          Value: "60"
```

# Register targets for your Network Load Balancer

When your target is ready to handle requests, you register it with one or more target groups. The target type of the target group determines how you register targets. For example, you can register instance IDs, IP addresses, or an Application Load Balancer. Your Network Load Balancer starts routing requests to targets as soon as the registration process completes and the targets pass 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 Network Load Balancer target groups.

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.

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*.

**Contents**

# Target security groups

Before adding targets to your target group, configure the security groups associated with the targets to accept traffic from your Network Load Balancer.

### Recommendations for target security groups if the load balancer has an associated security group

- **To allow client traffic**: Add a rule that references the security group associated with the load balancer.
- **To allow PrivateLink traffic**: If you configured the load balancer to evaluate inbound rules for traffic sent through AWS PrivateLink, add a rule that accepts traffic from the load balancer security group on the traffic port. Otherwise, add a rule that accepts traffic from the load balancer private IP addresses on the traffic port.
- **To accept load balancer health checks**: Add a rule that accepts health check traffic from the load balancer security groups on the health check port.

### Recommendations for target security groups if the load balancer is not associated with a security group

- **To allow client traffic**: If your load balancer preserves client IP addresses, add a rule that accepts traffic from the IP addresses of approved clients on the traffic port. Otherwise, add a rule that accepts traffic from the load balancer private IP addresses on the traffic port.
- **To allow PrivateLink traffic**: Add a rule that accepts traffic from the load balancer private IP addresses on the traffic port.
- **To accept load balancer health checks**: Add a rule that accepts health check traffic from the load balancer private IP addresses on the health check port.

### How client IP preservation works

Network Load Balancers don't preserve client IP addresses unless you set the `preserve_client_ip.enabled` attribute to `true`. Also, with dualstack Network Load Balancers, client IP address preservation does not work when translating IPv4 addresses to IPv6, or IPv6 to IPv4 addresses. Client IP address preservation only works when client and target IP addresses are both IPv4 or both IPv6.

**To find the load balancer private IP addresses using the console**

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, enter 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 **Private IPv4 address**.

For more information, see Update the security groups for your Network Load Balancer.

## 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* | *target port* | Allow client traffic (IP Preservation: ON) |
| *VPC CIDR* | *listener* | *target port* | Allow client traffic (IP Preservation: OFF) |
| *VPC CIDR* | *health check* | *health check* | Allow health check traffic |
| **Outbound** | | | |
| **Destination** | **Protocol** | **Port Range** | **Comment** |

| | | | |
|---|---|---|---|
| *Client IP addresses* | *listener* | 1024-65535 | Allow return traffic to client (IP Preservation: ON) |
| *VPC CIDR* | *listener* | 1024-65535 | Allow return traffic to client (IP Preservation: OFF) |
| *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 |
| *VPC CIDR* | *listener* | 1024-65535 | Allow response from target |
| *VPC CIDR* | *health check* | 1024-65535 | Allow health check traffic |
| **Outbound** | | | |
| **Destination** | **Protocol** | **Port Range** | **Comment** |
| *Client IP addresses* | *listener* | 1024-65535 | Allow responses to clients |
| *VPC CIDR* | *listener* | *target port* | Allow requests to targets |

| | | | |
|---|---|---|---|
| *VPC CIDR* | *health check* | *health check* | Allow health check to targets |

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.

## Shared subnets

Participants can create a Network Load Balancer in a shared VPC. Participants can't register a target that runs in a subnet that is not shared with them.

Shared subnets for Network Load Balancers is supported in all AWS Regions, excluding:

- Asia Pacific (Osaka) `ap-northeast-3`
- Asia Pacific (Hong Kong) `ap-east-1`
- Middle East (Bahrain) `me-south-1`
- AWS China (Beijing) `cn-north-1`
- AWS China (Ningxia) `cn-northwest-1`

## Register 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 which targets you can register. For more information, see Target type. Use the information below to register targets with a target group of type `instance` or `ip`. If the target type is `alb`, see Use Application Load Balancers as targets.

**Requirements and considerations**

- An instance must be in the `running` state when you register it.
- You can't 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.
- When registering targets by instance ID, instances must be in the same VPC as the Network Load Balancer. You can't register instances by instance ID if they are in an VPC that is peered to

the load balancer VPC (same Region or different Region). You can register these instances by IP address.

- When registering targets by instance ID for a IPv6 target group, the targets must have an assigned primary IPv6 address. To learn more, see IPv6 addresses in the *Amazon EC2 User Guide*
- When registering targets by IP address for an IPv4 target group, the IP addresses that you register must be from one of the following CIDR blocks:
  - The subnets of the target group VPC
  - 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)
- When registering targets by IP address for an IPv6 target group, the IP addresses that you register must be within the VPC IPv6 CIDR block or within the IPv6 CIDR block of a peered VPC.
- 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.

Console

**To register targets**

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. Choose **Register targets**.
6. If the target type of the target group is `instance`, select available instances, override the default port if needed, and then choose **Include as pending below**.
7. If the target type of the target group is `ip`, for each IP address, select the network, enter the IP address and ports, and choose **Include as pending below**.

8. If the target type of the target group is `alb`, override the default port if needed and select the Application Load Balancer. For more information, see Use Application Load Balancers as targets.
9. Choose **Register pending targets**.

AWS CLI

**To register targets**

Use the register-targets command. The following example registers targets by instance ID. Because the port is not specified, the load balancer uses the target group port.

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

The following example registers targets by IP address. Because the port is not specified, the load balancer uses the target group port.

```
aws elbv2 register-targets \
    --target-group-arn target-group-arn \
    --targets Id=10.0.50.10 Id=10.0.50.20
```

The following example registers an Application Load Balancer as a target.

```
aws elbv2 register-targets \
    --target-group-arn target-group-arn \
    --targets Id=application-load-balancer-arn
```

CloudFormation

**To register targets**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the new targets. The following example registers two targets by instance ID.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
```

```
      Name: my-target-group
      Protocol: HTTP
      Port: 80
      TargetType: instance
      VpcId: !Ref myVPC
      Targets:
        - Id: !GetAtt Instance1.InstanceId
          Port: 80
        - Id: !GetAtt Instance2.InstanceId
          Port: 80
```

## Deregister targets

If demand on your application decreases, or if 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.

Console

**To deregister targets**

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 **Targets** tab, select the targets to remove.
5. Choose **Deregister**.

AWS CLI

**To deregister targets**

Use the deregister-targets command. The following example deregisters two targets that were registered by instance ID.

```
aws elbv2 deregister-targets \
    --target-group-arn target-group-arn \
    --targets Id=i-1234567890abcdef0 Id=i-0abcdef1234567890
```

# Use an Application Load Balancer as a target of a Network Load Balancer

You can create a target group with a single Application Load Balancer as the target, and configure your Network Load Balancer to forward traffic to it. In this scenario, the Application Load Balancer takes over the load balancing decision as soon as traffic reaches it. This configuration combines the features of both load balancers and offers the following advantages:

- You can use the layer 7 request-based routing feature of the Application Load Balancer in combination with features that the Network Load Balancer supports, such as endpoint services (AWS PrivateLink) and static IP addresses.
- You can use this configuration for applications that need a single endpoint for multi-protocols, such as media services using HTTP for signaling and RTP to stream content.

You can use this feature with an internal or internet-facing Application Load Balancer as the target of an internal or internet-facing Network Load Balancer.

**Considerations**

- You can only register one Application Load Balancer per target group.
- To associate an Application Load Balancer as a target of a Network Load Balancer, the load balancers must be in the same VPC within the same account.
- You can associate an Application Load Balancer as a target of up to two Network Load Balancers. To do this, register the Application Load Balancer with a separate target group for each Network Load Balancer.
- Each Application Load Balancer that you register with a Network Load Balancer decreases the maximum number of targets per Availability Zone per Network Load Balancer by 50. You can disable cross-zone load balancing in both load balancers to minimize latency and avoid Regional data transfer charges. For more information, see Quotas for your Network Load Balancers.
- When the target group type is `alb`, you can't modify the target group attributes. These attributes always use their default values.
- After you register an Application Load Balancer as a target, you can't delete the Application Load Balancer until you deregister it from all target groups.
- The communication between a Network Load Balancer and an Application Load Balancer always uses IPv4.

**Tasks**



## Prerequisite

If you don't already have an Application Load Balancer to use as a target, create the load balancer, its listeners, and its target groups. For more information, see Create an Application Load Balancer in the *User Guide for Application Load Balancers*.

## Step 1: Create a target group of type alb

Create a target group of type `alb`. You can register your Application Load Balancer as a target when you create the target group or later on.

Console

**To create a target group for an Application Load Balancer as a target**

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 **Create target group**.
4. In the **Basic configuration** pane, for **Choose a target type**, choose **Application Load Balancer**.
5. For **Target group name**, enter a name for the target group.
6. For **Protocol**, only TCP is allowed. Select the **Port** for your target group. The port for this target group must match the listener port of the Application Load Balancer. If you choose a different port for this target group, you can update the listener port on the Application Load Balancer to match it.
7. For **VPC**, select the virtual private cloud (VPC) for the target group. This must be the same VPC used by the Application Load Balancer.
8. For **Health checks**, choose HTTP or HTTPS as the **Health check protocol**. Health checks are sent to the Application Load Balancer and forwarded to its targets using the specified port, protocol, and ping path. Ensure that your Application Load Balancer can receive these

health checks by having a listener with a port and protocol that matches the health check port and protocol.

9. (Optional) Expand **Tags**. For each tag, choose **Add new tag** and enter a tag key and a tag value.
10. Choose **Next**.
11. If you are ready to register the Application Load Balancer, choose **Register now**, override the default port if needed, and select the Application Load Balancer. The Application Load Balancer must have a listener on the same port as the target group. You can add or edit a listener on this load balancer to match the target group port, or return to the previous step and change the port for the target group.

    If you are not ready to register the Application Load Balancer as a target, choose **Register later** and register the target later on. For more information, see the section called "Register targets".
12. Choose **Create target group**.

AWS CLI

**To create a target group of type alb**

Use the create-target-group command. The protocol must be TCP and the port must match the listener port of the Application Load Balancer.

```
aws elbv2 create-target-group \
    --name my-target-group \
    --protocol TCP \
    --port 80 \
    --target-type alb \
    --vpc-id vpc-1234567890abcdef0 \
    --tags Key=department,Value=123
```

CloudFormation

**To create a target group of type alb**

Define a resource of type AWS::ElasticLoadBalancingV2::TargetGroup. The protocol must be TCP and the port must match the listener port of the Application Load Balancer.

```
Resources:
```

```
myTargetGroup:
  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
  Properties:
    Name: my-target-group
    Protocol: TCP
    Port: 80
    TargetType: alb
    VpcId: !Ref myVPC
    Tags:
      - Key: 'department'
        Value: '123'
    Targets:
      - Id: !Ref myApplicationLoadBalancer
        Port: 80
```

## Step 2: Create a Network Load Balancer and configure routing

When you create the Network Load Balancer, you can configure the default action to forward traffic to the Application Load Balancer.

Console

**To create the Network Load Balancer**

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. Under **Network Load Balancer**, choose **Create**.
5. **Basic configuration**
   a. For **Load balancer name**, enter a name for your Network Load Balancer.
   b. For **Scheme**, choose **Internet-facing** or **Internal**. An internet-facing Network Load Balancer routes requests from clients to targets over the internet. An internal Network Load Balancer routes requests to targets using private IP addresses.
   c. For **Load balancer IP address type**, choose **IPv4** if your clients use IPv4 addresses to communicate with the Network Load Balancer or **Dualstack** if your clients use both IPv4 and IPv6 addresses to communicate with the Network Load Balancer.
6. **Network mapping**

a. For **VPC**, select the same VPC that you used for your Application Load Balancer. With an internet-facing load balancer, only VPCs with an internet gateway are available for selection.

b. For **Availability Zones and subnets**, select at least one Availability Zones, and select one subnet per zone. We recommend that you select the same Availability Zones that are enabled for your Application Load Balancer. This optimizes availability, scaling, and performance.

   (Optional) To use static IP addresses, choose **Use an Elastic IP address** in the **IPv4 settings** for each Availability Zone. With static IP addresses you can add certain IP addresses to an allow list for firewalls, or you can hard code IP addresses with clients.

7. **Security groups**

   We preselect the default security group for the load balancer VPC. You can select additional security groups as needed. If you don't have a security group that meets your needs, choose **create a new security group** to create one now. For more information, see Create a security group in the *Amazon VPC User Guide*.

   > **Warning**
   >
   > If you don't associate any security groups with your Network Load Balancer now, you can't associate them later on.

8. **Listeners and routing**

   a. The default is a listener that accepts TCP traffic on port 80. Only TCP listeners can forward traffic to an Application Load Balancer target group. You must keep **Protocol** as **TCP**, but you can modify **Port** as needed.

      With this configuration, you can use HTTPS listeners on the Application Load Balancer to terminate TLS traffic.

   b. For **Default action**, select the target group that you created in the previous step.

   c. (Optional) Choose **Add listener tag** and enter a tag key and a tag value.

9. **Load balancer tags**

   (Optional) Expand **Load balancer tags**. Choose **Add new tag** and enter a tag key and a tag value. For more information, see Tags.

10. **Summary**

    Review your configuration and choose **Create load balancer**.

AWS CLI

**To create the Network Load Balancer**

Use the create-load-balancer command. We recommend that you use the same Availability Zones that are enabled for your Application Load Balancer.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --type network \
    --scheme internal \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890 \
    --security-groups sg-1111222233334444
```

**To add a TCP listener**

Use the create-listener command to add a TCP listener. Only TCP listeners can forward traffic to an Application Load Balancer. For the default action, use the target group that you created in the previous step.

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

CloudFormation

**To create the Network Load Balancer**

Define a resource of type AWS::ElasticLoadBalancingV2::LoadBalancer and a resource of type AWS::ElasticLoadBalancingV2::Listener. Only TCP listeners can forward traffic to an Application Load Balancer. For the default action, use the target group that you created in the previous step.

```
Resources:
  myLoadBalancer:
```

```
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-load-balancer
      Type: network
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup

  myTCPListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: TCP
      Port: 80
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

## Step 3: (Optional) Create a VPC endpoint service

To use the Network Load Balancer that you set up in the previous step as an endpoint for private connectivity, you can enable AWS PrivateLink. This establishes a private connection to your load balancer as an endpoint service.

**To create a VPC endpoint service using your Network Load Balancer**

1. On the navigation pane, choose **Load Balancers**.
2. Select the name of the Network Load Balancer to open its details page.
3. On the **Integrations** tab, expand **VPC Endpoint Services (AWS PrivateLink)**.
4. Choose **Create endpoint services** to open the **Endpoint services** page. For the remaining steps, see Create an endpoint service in the *AWS PrivateLink Guide*.

# Tag a target group for your Network Load Balancer

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.

Console

**To manage the tags for a target group**

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. Choose **Save changes**.

AWS CLI

**To add tags**

Use the add-tags command. The following example adds two tags.

```
aws elbv2 add-tags \
    --resource-arns target-group-arn \
    --tags "Key=project,value=lima" "Key=department,Value=digital-media"
```

**To remove tags**

Use the remove-tags command. The following example removes the tags with the specified keys.

```
aws elbv2 remove-tags \
    --resource-arns target-group-arn \
    --tag-keys project department
```

CloudFormation

**To add tags**

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `Tags` property.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: TCP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      Tags:
        - Key: 'project'
          Value: 'lima'
        - Key: 'department'
          Value: 'digital-media'
```

# Delete a target group for your Network Load Balancer

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.

Console

**To delete a target group**

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. Choose **Delete**.

AWS CLI

**To delete a target group**

Use the delete-target-group command.

```
aws elbv2 delete-target-group \
    --target-group-arn target-group-arn
```

# 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.

**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 group's client IP preservation attribute is enabled, the connection appears to the instance as a connection from the client. Otherwise, the connection's source IP is the load balancer's private IP address. 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.

If a Network Load Balancer has associated security groups, your flow logs contain entries for traffic that is allowed or rejected by the security groups. For Network Load Balancers with TLS listeners, your flow logs entries reflect only the rejected entries.

**Amazon CloudWatch Internet Monitor**

You can use Internet Monitor for visibility into how internet issues impact the performance and availability between your applications hosted on AWS and your end users. You can also explore,

in near real-time, how to improve the projected latency of your application by switching to use other services, or by rerouting traffic to your workload through different AWS Regions. For more information, see Using Amazon CloudWatch Internet Monitor.

**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.

**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 Log API calls for Elastic Load Balancing using CloudTrail.

# 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 Network Load Balancers with security groups, traffic rejected by the security groups is not captured in the CloudWatch metrics.

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**: Always reported.<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 connections in the SYN_SENT and 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`. |

| Metric | Description |
| --- | --- |
| | **Dimensions**<br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |
| ActiveFlowCount_TLS | The total number of concurrent TLS flows (or connections) from clients to targets. This metric includes connections in the SYN_SENT and ESTABLISHED state.<br>**Reporting criteria**: There is a nonzero value.<br>**Statistics**: The most useful statistics are Average, Maximum, and Minimum.<br>**Dimensions**<br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |
| ActiveFlowCount_UDP | The total number of concurrent UDP flows (or connections) from clients to targets.<br>**Reporting criteria**: There is a nonzero value.<br>**Statistics**: The most useful statistics are Average, Maximum, and Minimum.<br>**Dimensions**<br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |

| Metric | Description |
| --- | --- |
| ActiveZonalShiftHostCount | The number of targets that are actively participating in zonal shift currently.<br><br>**Reporting criteria**: Reported when the load balancer is opt-in for zonal shift.<br><br>**Statistics**: The most useful statistics are Maximum, and Minimum.<br><br>**Dimensions**<br><br>• LoadBalancer , TargetGroup<br>• AvailabilityZone , LoadBalancer , TargetGroup |
| 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 |
| 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**: There is a nonzero value.<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**: There is a nonzero value.<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**: There is a nonzero value.<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• LoadBalancer |

| Metric | Description |
| --- | --- |
| HealthyHostCount | The number of targets that are considered healthy. This metric does not include any Application Load Balancers registered as targets.<br><br>**Reporting criteria**: Reported if there are registered targets.<br><br>**Statistics**: The most useful statistics are Maximum and Minimum.<br><br>**Dimensions**<br><br>• LoadBalancer ,TargetGroup<br>• AvailabilityZone ,LoadBalancer ,TargetGroup |
| NewFlowCount | The total number of new flows (or connections) established from clients to targets in the time period.<br><br>**Reporting criteria**: Always reported.<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 |

| Metric | Description |
| --- | --- |
| 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 |
| PeakBytesPerSecond | The highest average bytes processed per second, calculated every 10 seconds during the sampling window. This metric does not include health check traffic.<br><br>**Reporting criteria**: Always reported<br><br>**Statistics**: The most useful statistic is Maximum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |

| Metric | Description |
| --- | --- |
| PeakPacketsPerSecond | Highest average packet rate (packets processed per second), calculated every 10 seconds during the sampling window. This metric includes health check traffic.<br><br>**Reporting criteria**: Always reported.<br><br>**Statistics**: The most useful statistic is Maximum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |
| PortAllocationErrorCount | The total number of ephemeral port allocation errors during a client IP translation operation. A non-zero value indicates dropped client connections.<br><br>Note: Network Load Balancers support 55,000 simultaneous connections or about 55,000 connections per minute to each unique target (IP address and port) when performing client address translation. To fix port allocation errors, add more targets to the target group.<br><br>**Reporting criteria**: Always reported.<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**: Always reported.<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 |

| Metric | Description |
| --- | --- |
| 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 |
| ProcessedPackets | The total number of packets processed by the load balancer. This count includes traffic to and from targets, including health check traffic.<br><br>**Reporting criteria**: Always reported.<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |
| RejectedFlowCount | The total number of flows (or connections) rejected by the load balancer.<br><br>**Reporting criteria**: Always reported.<br><br>**Statistics**: The most useful statistics are Average, Maximum, and Minimum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |

| Metric | Description |
| --- | --- |
| RejectedFlowCount_TCP | The number of TCP flows (or connections) rejected 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 |
| ReservedLCUs | The number of load balancer capacity units (LCUs) reserved for your load balancer using LCU Reservation.<br><br>**Reporting criteria**: There is a nonzero value<br><br>**Statistics**: All<br><br>**Dimensions**<br><br>• LoadBalancer |
| SecurityGroupBlockedFlowCount_Inbound_ICMP | The number of new ICMP messages rejected by the inbound rules of the load balancer security groups.<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 |
| --- | --- |
| SecurityGroupBlockedFlowCount_Inbound_TCP | The number of new TCP flows rejected by the inbound rules of the load balancer security groups.<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 |
| SecurityGroupBlockedFlowCount_Inbound_UDP | The number of new UDP flows rejected by the inbound rules of the load balancer security groups.<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 |
| SecurityGroupBlockedFlowCount_Outbound_ICMP | The number of new ICMP messages rejected by the outbound rules of the load balancer security groups.<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 |
| --- | --- |
| SecurityGroupBlockedFlowCount_Outbound_TCP | The number of new TCP flows rejected by the outbound rules of the load balancer security groups.<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 |
| SecurityGroupBlockedFlowCount_Outbound_UDP | The number of new UDP flows rejected by the outbound rules of the load balancer security groups.<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 |

| 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**: Always reported.<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. For more information, see Troubleshooting.<br><br>**Reporting criteria**: Always reported.<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**: Always reported.<br><br>**Statistics**: The most useful statistic is Sum.<br><br>**Dimensions**<br><br>• LoadBalancer<br>• AvailabilityZone ,LoadBalancer |

| Metric | Description |
| --- | --- |
| UnHealthyHostCount | The number of targets that are considered unhealthy. This metric does not include any Application Load Balancers registered as targets.<br><br>**Reporting criteria**: Reported if there are registered targets.<br><br>**Statistics**: The most useful statistics are Maximum and Minimum.<br><br>**Dimensions**<br><br>• LoadBalancer , TargetGroup<br>• AvailabilityZone , LoadBalancer , TargetGroup |
| UnhealthyRoutingFlowCount | The number of flows (or connections) that are routed using the routing failover action (fail open). This metric is not supported for TLS listeners.<br><br>**Reporting criteria**: There is a nonzero value.<br><br>**Statistics**: The most useful statistic is Sum. |
| ZonalHealthStatus | The number of Availability Zones that the load balancer considers healthy. The load balancer emits a 1 for each healthy Availability Zone and a 0 for each unhealthy Availability Zone.<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<br>• AvailabilityZone , LoadBalancer |

## 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`. It's recommended to monitor maximum `HealthyHostCount`, invoking the alarm when the maximum `HealthyHostCount` falls below your required minimum, or being `0`. This can help in identifying when your targets have become unhealthy. It's also recommended to monitor minimum `UnHealthyHostCount`, invoking the alarm when the minimum `UnHealthyHostCount` rises above `0`. This allows you to become aware when there are no longer any registered targets.

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 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 connections established with 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 the logs contain information about TLS requests only. Access logs record requests on a best-effort basis. We recommend that you use access logs to understand the nature of the requests, not as a complete accounting of all 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.

You can enable server-side encryption with Amazon S3-managed encryption keys (SSE-S3), or using Key Management Service with Customer Managed Keys (SSE-KMS CMK) 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 KMS key that is regularly rotated. For more information, see Specifying Amazon S3 encryption (SSE-S3) and Specifying server-side encryption with AWS KMS (SSE-KMS) in the *Amazon S3 User 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.

**Contents**

# 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_net.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.

The following is an example log file name:

```
s3://my-bucket/prefix/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2020/05/01/123456789012_elasticloadbalancing_us-east-2_net.my-
loadbalancer.1234567890abcdef_20200501T0000Z_20sg8hgm.log.gz
```

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 Manage your storage lifecycle in the *Amazon S3 User 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. |

| Field | Description |
| --- | --- |
| 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. If there is no TLS handshake or a TLS handshake failure, this value is set to `-`. |
| 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`, `tlsv12`, and `tlsv13`. If TLS negotiation does not complete, this value is set to `-`. |
| 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 `-`. |

| Field | Description |
| --- | --- |
| 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. |
| tls_connection_creation_time | The time recorded at the beginning of the TLS connection, in ISO 8601 format. |

## 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
- - - 2018-12-20T02:59:30
```

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" 2020-04-01T08:51:20
```

## 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

## Enable access logs for your Network Load Balancer

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. The bucket must have a bucket policy that grants Elastic Load Balancing permission to write to the bucket.

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

### Bucket requirements

You can use an existing bucket, or create a bucket specifically for access logs. The bucket must meet the following requirements.

### Requirements

- The bucket must be located in the same Region as the load balancer. The bucket and the load balancer can be owned by different accounts.
- 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.
- 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.

### Example bucket policy

The following is an example policy. For the `Resource` elements, replace *`amzn-s3-demo-destination-bucket`* with the name of the S3 bucket for your access logs. Be sure to omit the *`Prefix/`* if you are not using a bucket prefix. For `aws:SourceAccount`, specify the ID of the AWS account with the load balancer. For `aws:SourceArn`, replace *`region`* and *`012345678912`* with the Region and account ID of the load balancer, respectively.

JSON

```
{
    "Version": "2012-10-17",
    "Id": "AWSLogDeliveryWrite",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryAclCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "012345678912"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
```

```
                        "arn:aws:logs:us-east-1:012345678912:*"
                    ]
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-
bucket/Prefix/AWSLogs/account-ID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": [
                        "012345678912"
                    ]
                },
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:logs:us-east-1:012345678912:*"
                    ]
                }
            }
        }
    ]
}
```

### Encryption

You can enable server-side encryption for your Amazon S3 access log bucket in one of the following ways:

- Amazon S3-Managed Keys (SSE-S3)
- AWS KMS keys stored in AWS Key Management Service (SSE-KMS) †

† With Network Load Balancer access logs, you can't use AWS managed keys, you must use customer managed keys.

For more information, see Specifying Amazon S3 encryption (SSE-S3) and Specifying server-side encryption with AWS KMS (SSE-KMS) in the *Amazon S3 User Guide*.

The key policy must allow the service to encrypt and decrypt the logs. The following is an example policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

## Configure access logs

Use the following procedure to configure access logs to capture request information and deliver log files to your S3 bucket.

Console

**To enable access logs**

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

2. In the navigation pane, choose **Load Balancers**.
3. Select the name of your load balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. For **Monitoring**, turn on **Access logs**.
6. For **S3 URI**, enter the S3 URI for your log files. The URI that you specify depends on whether you're using a prefix.
   - URI with a prefix: s3://*amzn-s3-demo-logging-bucket*/*logging-prefix*
   - URI without a prefix: s3://*amzn-s3-demo-logging-bucket*
7. Choose **Save changes**.

AWS CLI

**To enable access logs**

Use the modify-load-balancer-attributes command with the related attributes.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes \
        Key=access_logs.s3.enabled,Value=true \
        Key=access_logs.s3.bucket,Value=amzn-s3-demo-logging-bucket \
        Key=access_logs.s3.prefix,Value=logging-prefix
```

CloudFormation

**To enable access logs**

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the related attributes.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-nlb
      Type: network
      Scheme: internal
      Subnets:
```

```
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "access_logs.s3.enabled"
          Value: "true"
        - Key: "access_logs.s3.bucket"
          Value: "amzn-s3-demo-logging-bucket"
        - Key: "access_logs.s3.prefix"
          Value: "logging-prefix"
```

## Disable access logs for your Network Load Balancer

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 Creating, configuring, and working with S3 buckets in the *Amazon S3 User Guide*.

Console

**To disable access logs**

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the name of your load balancer to open its details page.
4. On the **Attributes** tab, choose **Edit**.
5. For **Monitoring**, turn off **Access logs**.
6. Choose **Save changes**.

AWS CLI

**To disable access logs**

Use the modify-load-balancer-attributes command.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes Key=access_logs.s3.enabled,Value=false
```

# 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 Network Load Balancer target groups.

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. Also, the security group for your load balancer must allow traffic to the instances. For more information, see Update the security groups for your Network Load Balancer.

**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.

## 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. Also, the security group for your load balancer must allow traffic to the instances. For more information, see Update the security groups for your Network Load Balancer.

**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.

**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

This occurs when all registered targets are unhealthy. If there is at least one healthy registered target, your Network Load Balancer routes requests only to its healthy registered targets.

When there are only unhealthy registered targets, the Network Load Balancer routes requests to all the registered targets, known as fail-open mode. The Network Load Balancer does this instead of removing all the IP addresses from DNS when all the targets are unhealthy and respective Availability Zones do not have healthy target to send request to.

# 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.

# Unable to associate a security group with a load balancer

If the Network Load Balancer was created without security groups, it can't support security groups after creation. You can only associate a security group to a load balancer during creation, or to an existing load balancer that was originally created with security groups.

# Unable to remove all security groups

If the Network Load Balancer was created with security groups, there must be at least one security group associated with it at all times. You cannot remove all security groups from the load balancer at the same time.

# Increase in TCP_ELB_Reset_Count metric

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 the 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. Additionally, if a target becomes unhealthy, the load balancer sends a TCP RST for packets received on the client connections associated with the target, unless the unhealthy target triggers the load balancer to fail open.

If you see a spike in the `TCP_ELB_Reset_Count` metric just before or just as the `UnhealthyHostCount` metric increases, it is likely that the TCP RST packets were sent because the target was starting to fail but hadn't been marked unhealthy. If you see persistent increases in `TCP_ELB_Reset_Count` without targets being marked unhealthy, you can check the VPC flow logs for clients sending data on expired flows.

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

Check whether client IP preservation is enabled on your target group. NAT loopback, also known as hairpinning, is not supported when client IP preservation is enabled.

If an instance is a client of a load balancer that it's registered with and it has client IP preservation enabled, the connection succeeds only if the request is routed to a different instance. If the request is routed to the same instance it was sent from, the connection times out because the source and destination IP addresses are the same. Note that this applies to Amazon EKS pods running in the same EC2 worker node instance, even though they have different IP addresses.

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

- Disable client IP preservation. Instead, use Proxy Protocol v2 to get the client IP address.
- Ensure that containers that must communicate are on different container instances.

## 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 for backend flows

With PrivateLink traffic or when client IP preservation is disabled, a Network Load Balancer supports 55,000 simultaneous connections or about 55,000 connections per minute to each unique target (IP address and port). If you exceed these limits, there is an increased chance of port allocation errors. You can track port allocation errors using the `PortAllocationErrorCount` metric. You can track active connections using the `ActiveFlowCount` metric. For more information, see CloudWatch metrics for your Network Load Balancer.

To fix port allocation errors, we recommend that you add targets to the target group.

Alternatively, if you can't add targets to the target group, you can add up to 7 secondary IP addresses to the load balancer network interfaces. The secondary IP addresses are automatically allocated from the IPv4 CIDR blocks of the corresponding subnets. Each secondary IP address consumes 6 network addressing units. Note that after you add a secondary IP address you can't remove it. The only way to release the secondary IP addresses is to delete the load balancer.

## Intermittent TCP connection establishment failure or TCP connection establishment delays

When client IP address preservation is enabled, a client may connect to different destination IP address using the same source ephemeral port. These destination IP addresses can be from the same load balancer (in different Availability Zones) when cross-zone load balancing enabled or different Network Load Balancers that uses the same target IP address and port registered. In this case, if these connections are routed to the same target IP address and port, the target will see a duplicated connection, since they come from the same client IP address and port. This leads to connection errors and delays when establishing one of these connections. This occurs frequently when a NAT device in front of the client, and the same source IP address and source port is allocated when connecting to multiple Network Load Balancer IP addresses simultaneously.

You can reduce this type of connection error by increasing the number of source ephemeral ports allocated by the client or NAT device, or by increasing the number of targets for the load balancer. We recommend clients change the source port used when reconnecting after these connection failures. To prevent this type of connection error, if you are using a single Network Load Balancer, you can consider disabling cross-zone load balancing, or if using multiple Network Load Balancers, you can consider not using the same target IP address and port registered in multiple target groups. Alternatively, you can consider disabling client IP preservation. If you need the client IP you can use retrieve it using Proxy Protocol v2. To learn more about Proxy Protocol v2, see Proxy protocol.

## Potential failure when the load balancer is being provisioned

One of the reasons a Network Load Balancer could fail when it is being provisioned is if you use an IP address that is already assigned or allocated elsewhere (for example, assigned as a secondary IP address for an EC2 instance). This IP address prevents the load balancer from being set up, and its state is `failed`. You can resolve this by de-allocating the associated IP address and retrying the creation process.

# Traffic is distributed unevenly between targets

TCP and TLS listeners route TCP connections and UDP listeners route UDP streams. The load balancer selects targets using a flow hash algorithm. A single connection from a client is inherently sticky.

If you notice that some targets appear to receive more traffic than others, we recommend that you review the VPC flow logs. Compare the number of unique connections for each target IP address. Keep the time window as short as possible, as target registration, deregistration, and unhealthy targets influence these connection numbers.

The following are possible scenarios where connections can be distributed unevenly:

- If you start with a small number of targets and then register additional targets later on, the original targets still have connections with clients. With an HTTP workload, keepalives ensure that clients reuse connections. If you lower the max keepalives on your web application, clients would open new connections more often.
- If target group stickiness is enabled, there is a small number of clients, and the clients communicate through a NAT device with a single source IP address, connections from these clients are routed to the same target.
- If cross-zone load balancing is disabled and clients prefer the load balancer IP address from one of the load balancer zones, connections would be distributed unevenly between the load balancer zones.

# DNS name resolution contains fewer IP addresses than enabled Availability Zones

Ideally your Network Load Balancer provides one IP address per enabled Availability Zone, when they have at least one healthy host in the Availability Zone. When there are no healthy host in a particular Availability Zone, and cross-zone load balancing is disabled, the IP address of the Network Load Balancer respective of that AZ will be removed from DNS.

For example, suppose your Network Load Balancer has three Availability Zones enabled, all of which have at least one healthy registered target instance.

- If the registered target instance(s) in Availability Zone A become unhealthy, the corresponding IP address of Availability Zone A for the Network Load Balancer is removed from DNS.

- If any two of the enabled Availability Zones have no healthy registered target instance(s), the respective two IP addresses of the Network Load Balancer will be removed from DNS.
- If there are no healthy registered target instance(s) in all the enabled Availability Zones, the fail-open mode is enabled and DNS will provide all the IP addresses from the three enabled AZs in the result.

## IP fragmented packets are not routed to targets

Network Load Balancers do not support IP fragmented packets for non-UDP traffic.

## Troubleshoot unhealthy targets using the resource map

If your Network Load Balancer targets are failing health checks, you can use the resource map to find unhealthy targets and take actions based on the failure reason code. For more information, see View the Network Load Balancer resource map.

Resource map provides two views: **Overview**, and **Unhealthy Target Map**. **Overview** is selected by default and displays all of your load balancer's resources. Selecting the **Unhealthy Target Map** view will display only the unhealthy targets in each target group associated to the Network Load Balancer.

> **Note**
>
> **Show resource details** must be enabled to view the health check summary and error messages for all applicable resources within the resource map. When not enabled, you must select each resource to view its details.

The **Target groups** column displays a summary of the healthy and unhealthy targets for each target group. This can help determine if all the targets are failing health checks, or only specific targets are failing. If all targets in a target group are failing health checks, check the target group's health check settings. Select a target group's name to open its detail page in a new tab.

The **Targets** column displays the TargetID and the current health check status for each target. When a target is unhealthy, the health check failure reason code is displayed. When a single target is failing a health check, verify the target has sufficient resources. Select a target's ID to open its detail page in a new tab.

Selecting **Export** gives you the option of exporting the current view of your Network Load Balancer's resource map as a PDF.

Verify that your instance is failing health checks and then based on the failure reason code check for the following issues:

- **Unhealthy: Request timed out**
  - Verify the security groups and network access control lists (ACL) associated with your targets and Network Load Balancer are not blocking connectivity.
  - Verify the target has sufficient capacity available to accept connections from the Network Load Balancer.
  - The Network Load Balancer's health check responses can be viewed in each target's application logs. For more information, see Health check reason codes.
- **Unhealthy: FailedHealthChecks**
  - Verify the target is listening for traffic on the health check port.

    **When using a TLS listener**

    You choose which security policy is used for front-end connections. The security policy used for back-end connections is automatically selected based on the front-end security policy in use.

    - If your TLS listener is using a TLS 1.3 security policy for front-end connections, the `ELBSecurityPolicy-TLS13-1-0-2021-06` security policy is used for back-end connections.
    - If your TLS listener is not using a TLS 1.3 security policy for front-end connections, the `ELBSecurityPolicy-2016-08` security policy is used for back-end connections.

    For more information, see Security policies.
  - Verify the target is providing a server certificate and key in the correct format specified by the security policy.
  - Verify the target supports one or more matching ciphers, and a protocol provided by the Network Load Balancer to establish TLS handshakes.

# 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, submit a request for a service quota increase.

**Quotas**

- Load balancer
- Target groups
- Load Balancer Capacity Units

## Load balancer

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

| Name | Default | Adjustable |
|---|---|---|
| Certificates per Network Load Balancer | 25 | Yes |
| Listeners per Network Load Balancer | 50 | No |
| Network Load Balancer ENIs per VPC | 1,200 [1] | Yes |
| Network Load Balancers per Region | 50 | Yes |
| Targets per Availability Zone per Network Load Balancer | 500 [2, 3] | Yes |
| Targets per Network Load Balancer | 3,000 [3] | Yes |

[1] Each Network Load Balancer uses one network interface per zone. The quota is set at the VPC level. When sharing subnets or VPCs, the usage is calculated across all tenants.

[2] If a target is registered with *N* target groups, it counts as *N* targets toward this limit. Each Application Load Balancer that is a target of the Network Load Balancer counts as 50 targets if cross-zone load balancing is disabled or 100 targets if cross-zone load balancing is enabled.

[3] If cross-zone load balancing is enabled, the maximum is 500 targets per load balancer, regardless of the number of Availability Zones.

## Target groups

The following quotas are for target groups.

| Name | Default | Adjustable |
|---|---|---|
| Target Groups per Region | 3,000 [1] | Yes |
| Targets per Target Group per Region (instances or IP addresses) | 1,000 | Yes |
| Targets per Target Group per Region (Application Load Balancers) | 1 | No |

[1] This quota is shared by Application Load Balancers and Network Load Balancers.

## Load Balancer Capacity Units

The following quotas are for Load Balancer Capacity Units (LCUs).

| Name | Default | Adjustable |
|---|---|---|
| Reserved Network Load Balancer Capacity Units (LCUs) per Network Load Balancer, per availability zone | 45000 | Yes |
| Reserved Network Load Balancer Capacity Units (LCU) per Region | 0 | Yes |

# Document history for Network Load Balancers

The following table describes the releases for Network Load Balancers.

| Change | Description | Date |
| --- | --- | --- |
| Secondary IPv4 addresses | This release adds support to add secondary IPv4 addresses to the load balancer network interfaces. | July 29, 2025 |
| Disable Availability Zones | This releases adds support to disable an Availability Zone for an existing load balancer. | February 13, 2025 |
| Capacity Unit reservation | This release adds support to set a minimum capacity for your load balancer. | November 20, 2024 |
| UDP support over IPv6 for dualstack load balancers | This release enables clients to access UDP-based applicati ons using IPv6. | October 31, 2024 |
| RSA 3072-bit and ECDSA 256/384/521-bit certificates | This release adds support for RSA 3072-bit certifica tes, and Elliptic Curve Digital Signature Algorithm (ECDSA) 256, 384 and 521-bit certifica tes via AWS Certificate Manager (ACM). | January 19, 2024 |
| FIPS 140-3 TLS termination | This release adds security policies that use FIPS 140-3 crypotographic modules when terminating TLS connections. | November 20, 2023 |

| | | |
|---|---|---|
| Zonal DNS affinity | This release adds support for clients resolving the load balancer DNS to receive an IP address in the same Availability Zone (AZ) they are in. | October 12, 2023 |
| Disable unhealthy target connection termination | This release adds support to maintain active connections to targets that fail health checks. | October 12, 2023 |
| Default UDP connection termination | This release adds support to terminate UDP connections at the end of the deregistration timeout by default. | October 12, 2023 |
| Register targets using IPv6 | This release adds support to register instances as targets when addressed by IPv6. | October 2, 2023 |
| Security groups for your Network Load Balancer | This release adds support to associate security groups with your Network Load Balancers at creation. | August 10, 2023 |
| Target group health | This release adds support to configure the minimum count or percentage of targets that must be healthy, and what actions the load balancer takes when the threshold is not met. | November 17, 2022 |
| Health check configuration | This release provides improvements to health check configuration. | November 17, 2022 |

| | | |
|---|---|---|
| Cross-zone load balancing | This release adds support to configure cross-zone load balancing at the target group level. | November 17, 2022 |
| IPv6 target groups | This release adds support to configure IPv6 target groups for Network Load Balancers. | November 23, 2021 |
| IPv6 internal load balancers | This release adds support to configure IPv6 target groups for Network Load Balancers. | November 23, 2021 |
| TLS 1.3 | This release adds security policies supporting TLS version 1.3. | October 14, 2021 |
| Application Load Balancers as targets | This release adds support to configure an Application Load Balancer as the target of a Network Load Balancer. | September 27, 2021 |
| Client IP preservation | This release adds support to configure client IP preservat ion. | February 4, 2021 |
| 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 support 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 | 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 | This release adds support for specifying subnets that were shared with you by another AWS account. | November 26, 2019 |
| Private IP addresses | 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 | This release adds support for enabling additional Availability Zones after you create your load balancer. | November 25, 2019 |
| Security policies for FS | This release adds support for three additional predefined forward secrecy security policies. | October 8, 2019 |

| | | |
|---|---|---|
| SNI support | This release adds support for Server Name Indication (SNI). | September 12, 2019 |
| UDP protocol | This release adds support for the UDP protocol. | June 24, 2019 |
| Available in new region | This release adds support for Network Load Balancers in the Asia Pacific (Osaka) Region. | June 12, 2019 |
| TLS protocol | This release adds support for the TLS protocol. | January 24, 2019 |
| Cross-zone load balancing | 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 | This release of Elastic Load Balancing introduces Network Load Balancers. | September 7, 2017 |