aws

Application Load Balancers

ELB

ELB: Application Load Balancers

This documentation is a draft for private preview for regions in the AWS European Sovereign Cloud. Documentation content will continue to evolve. Published: December 30, 2025.

Table of Contents

What is an Application Load Balancer?

ELB 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. ELB scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

ELB 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 Application Load Balancers. For more information about the other load balancers, see the User Guide for Network Load Balancers, the User Guide for Gateway Load Balancers, and the User Guide for Classic Load Balancers.

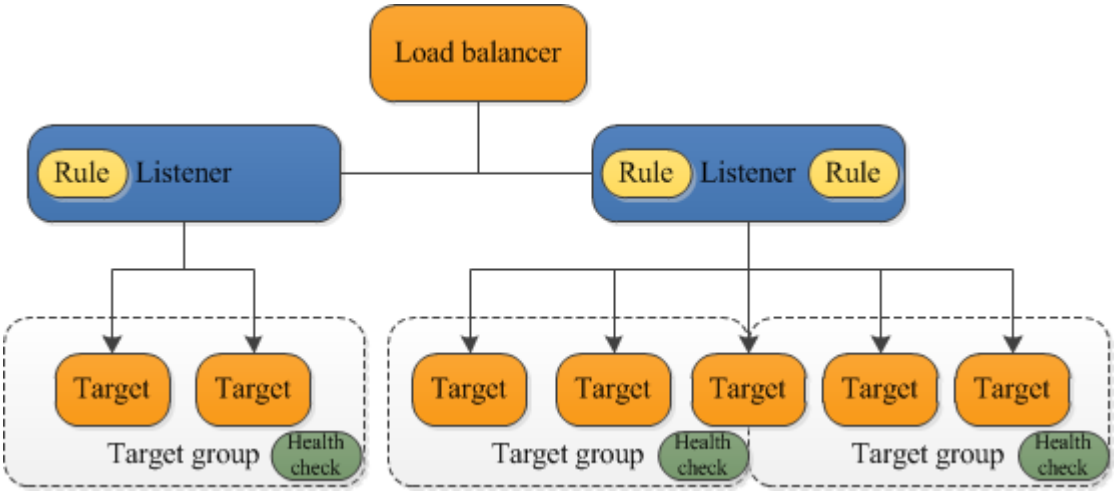
Application Load Balancer components

A *load balancer* serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. 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. The rules that you define for a listener determine how the load balancer routes requests to its registered targets. Each rule consists of a priority, one or more actions, and one or more conditions. When the conditions for a rule are met, then its actions are performed. You must define a default rule for each listener, and you can optionally define additional rules.

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

The following diagram illustrates the basic components. Notice that each listener contains a default rule, and one listener contains another rule that routes requests to a different target group. One target is registered with two target groups.



For more information, see the following documentation:

- Load balancers
- Listeners
- Target groups

Application Load Balancer overview

An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply, and then selects a target from the target group for the rule action. You can configure listener rules to route requests to different target groups based on the content of the application traffic. Routing is performed independently for each target group, even when a target is registered with multiple target groups. You can configure the routing algorithm used at the target group level. The default routing algorithm is round robin; alternatively, you can specify the least outstanding requests routing algorithm.

You can add and remove targets from your load balancer as your needs change, without disrupting the overall flow of requests to your application. ELB scales your load balancer as traffic to your application changes over time. ELB 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 ELB works in the *Elastic Load Balancing User Guide*.

Benefits of migrating from a Classic Load Balancer

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

- Support for Path conditions. You can configure rules for your listener that forward requests based on the URL in the request. This enables you to structure your application as smaller services, and route requests to the correct service based on the content of the URL.
- Support for Host conditions. You can configure rules for your listener that forward requests based on the host field in the HTTP header. This enables you to route requests to multiple domains using a single load balancer.
- Support for routing based on fields in the request, such as HTTP header conditions and methods, query parameters, and source IP addresses.
- Support for routing requests to multiple applications on a single EC2 instance. You can register an instance or IP address with multiple target groups, each on a different port.
- Support for redirecting requests from one URL to another.
- Support for returning a custom HTTP response.
- Support for registering targets by IP address, including targets outside the VPC for the load balancer.
- Support for registering Lambda functions as targets.
- Support for the load balancer to authenticate users of your applications through their corporate or social identities before routing requests.
- 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 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.
- Access logs contain additional information and are stored in compressed format.
- Improved load balancer performance.

Related services

ELB works with the following services to improve the availability and scalability of your applications.

- **Amazon EC2** — Virtual servers that run your applications in the cloud. You can configure your load balancer to route traffic to your EC2 instances.
- **Amazon EC2 Auto Scaling** — Ensures that you are running your desired number of instances, even if an instance fails, and enables you to automatically increase or decrease the number of instances as the demand on your instances changes. If you enable Auto Scaling with ELB, instances that are launched by Auto Scaling are automatically registered with the target group, and instances that are terminated by Auto Scaling are automatically de-registered from the target group.
- **AWS Certificate Manager** — When you create an HTTPS listener, you can specify certificates provided by ACM. The load balancer uses certificates to terminate connections and decrypt requests from clients. For more information, see SSL certificates for your Application Load Balancer.
- **Amazon CloudWatch** — Enables you to monitor your load balancer and take action as needed. For more information, see CloudWatch metrics for your Application Load Balancer.
- **Amazon ECS** — Enables you to run, stop, and manage Docker containers on a cluster of EC2 instances. You can configure your load balancer to route traffic to your containers. For more information, see Service load balancing in the *Amazon Elastic Container Service Developer Guide*.
- **AWS Global Accelerator** — Improves the availability and performance of your application. Use an accelerator to distribute traffic across multiple load balancers in one or more AWS Regions. For more information, see the AWS Global Accelerator Developer Guide.
- **Route 53** — Provides a reliable and cost-effective way to route visitors to websites by translating domain names (such as `www.example.com`) into the numeric IP addresses (such as `192.0.2.1`) that computers use to connect to each other. AWS assigns URLs to your resources, such as load balancers. However, you might want a URL that is easy for users to remember. For example, you can map your domain name to a load balancer. For more information, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.
- **AWS WAF** — You can use AWS WAF with your Application Load Balancer to allow or block requests based on the rules in a web access control list (web ACL). For more information, see AWS WAF.

To view information about services that are integrated with your load balancer, select your load balancer in the AWS Management Console and choose the **Integrated services** tab.

Pricing

With your load balancer, you pay only for what you use. For more information, see ELB pricing.

Application Load Balancers

A *load balancer* serves as the single point of contact for clients. Clients send requests to the load balancer, and the load balancer sends them to targets, such as EC2 instances. To configure your load balancer, you create target groups, and then register targets with your target groups. You also create listeners to check for connection requests from clients, and listener rules to route requests from clients to the targets in one or more target groups.

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

Contents

- Capacity reservations for your Application Load Balancer
- Integrations for your Application Load Balancer

Subnets for your load balancer

When you create an Application Load Balancer, you must enable the zones that contain your targets. To enable a zone, specify a subnet in the zone. ELB creates a load balancer node in each zone that you specify.

Considerations

- Your load balancer is most effective when you ensure that each enabled zone has at least one registered target.
- If you register targets in a zone but do not enable the zone, these registered targets do not receive traffic from the load balancer.
- If you enable multiple zones for your load balancer, the zones must be of the same type. For example, you can't enable both an Availability Zone and a Local Zone.
- You can specify a subnet that was shared with you.
- ELB creates network interfaces in the subnets where you configured your load balancer. These network interfaces are reserved so that the load balancer can complete maintenance actions even when the subnet is running low on available IP addresses. They have the description "ELI reserved by ELB for subnet".

Application Load Balancers support the following types of subnets.

Subnet types

- Availability Zone subnets
- Local Zone subnets
- Outpost subnets

Availability Zone subnets

You must select at least two Availability Zone subnets. The following restrictions apply:

- Each subnet must be from a different Availability Zone.

- To ensure that your load balancer can scale properly, verify that each Availability Zone subnet for your load balancer has a CIDR block with at least a `/27` bitmask (for example, `10.0.0.0/27`) and at least eight free IP addresses per subnet. These eight IP addresses are required to allow the load balancer to scale out if needed. Your load balancer uses these IP addresses to establish connections with the targets. Without them your Application Load Balancer could experience difficulties with node replacement attempts, causing it to enter a failed state.

  Note: If an Application Load Balancers subnet runs out of usable IP addresses while attempting to scale, the Application Load Balancer will run with insufficient capacity. During this time, old nodes continue to serve traffic, but the stalled scaling attempt might cause 5xx errors or timeouts when attempting to establish a connection.

Local Zone subnets

You can specify Local Zone subnets. The following features are not supported with local zone subnets:

- Lambda functions as targets
- Mutual TLS authentication
- AWS WAF integration

Outpost subnets

You can specify a single Outpost subnet. The following restrictions apply:

- You must have installed and configured an Outpost in your on-premises data center. You must have a reliable network connection between your Outpost and its AWS Region. For more information, see the AWS Outposts User Guide.
- The load balancer requires two `large` instances on the Outpost for the load balancer nodes. The supported instance types are shown in the following table. The load balancer scales as needed, resizing the nodes one size at a time (from `large` to `xlarge`, then `xlarge` to `2xlarge`, and then `2xlarge` to `4xlarge`). After scaling the nodes to the largest instance size, if you need additional capacity, the load balancer adds `4xlarge` instances as load balancer nodes. If you do not have sufficient instance capacity or available IP addresses to scale the load balancer, the load balancer reports an event to the AWS Health Dashboard and the load balancer state is `active_impaired`.

- You can register targets by instance ID or IP address. If you register targets in the AWS Region for the Outpost, they are not used.
- The following features are not supported:
  - AWS Global Accelerator integration
  - Lambda functions as targets
  - Mutual TLS authentication
  - Sticky sessions
  - User authentication
  - AWS WAF integration

An Application Load Balancer can be deployed on c5/c5d, m5/m5d, or r5/r5d instances on an Outpost. The following table shows the size and EBS volume per instance type that the load balancer can use on an Outpost:

Instance type and size	EBS volume (GB)
c5/c5d	
large	50
xlarge	50
2xlarge	50
4xlarge	100
m5/m5d	
large	50
xlarge	50
2xlarge	100
4xlarge	100
r5/r5d	

Instance type and size	EBS volume (GB)
large	50
xlarge	100
2xlarge	100
4xlarge	100

Load balancer security groups

A *security group* acts as a firewall that controls the traffic allowed to and from your load balancer. You can choose the ports and protocols to allow for both inbound and outbound traffic.

The rules for the security groups that are associated with your load balancer must allow traffic in both directions on both the listener and the health check ports. Whenever you add a listener to a load balancer or update the health check port for a target group, you must review your security group rules to ensure that they allow traffic on the new port in both directions. For more information, see Recommended rules.

Load balancer state

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

`provisioning`

The load balancer is being set up.

`active`

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

`active_impaired`

The load balancer is routing traffic but does not have the resources it needs to scale.

`failed`

The load balancer could not be set up.

Load balancer attributes

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

The following are the load balancer attributes:

`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 Enable access logs.

`access_logs.s3.prefix`

The prefix for the location in the Amazon S3 bucket.

`client_keep_alive.seconds`

The client keepalive value, in seconds. The default is 3600 seconds.

`deletion_protection.enabled`

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

`idle_timeout.timeout_seconds`

The idle timeout value, in seconds. The default is 60 seconds.

`ipv6.deny_all_igw_traffic`

Blocks internet gateway (IGW) access to the load balancer, preventing unintended access to your internal load balancer through an internet gateway. It is set to `false` for internet-facing load balancers and `true` for internal load balancers. This attribute does not prevent non-IGW internet access (such as, through peering, Transit Gateway, AWS Direct Connect, or Site-to-Site VPN).

`routing.http.desync_mitigation_mode`

Determines how the load balancer handles requests that might pose a security risk to your application. The possible values are `monitor`, `defensive`, and `strictest`. The default is `defensive`.

`routing.http.drop_invalid_header_fields.enabled`

Indicates whether HTTP headers with header fields that are not valid are removed by the load balancer (`true`), or routed to targets (`false`). The default is `false`. ELB requires that valid HTTP header names conform to the regular expression `[-A-Za-z0-9]+`, as described in the HTTP Field Name Registry. Each name consists of alphanumeric characters or hyphens. Select `true` if you want HTTP headers that do not conform to this pattern, to be removed from requests.

`routing.http.preserve_host_header.enabled`

Indicates whether the Application Load Balancer should preserve the `Host` header in the HTTP request and send it to targets without any change. The possible values are `true` and `false`. The default is `false`.

`routing.http.x_amzn_tls_version_and_cipher_suite.enabled`

Indicates whether the two headers (`x-amzn-tls-version` and `x-amzn-tls-cipher-suite`), which contain information about the negotiated TLS version and cipher suite, are added to the client request before sending it to the target. The `x-amzn-tls-version` header has information about the TLS protocol version negotiated with the client, and the `x-amzn-tls-cipher-suite` header has information about the cipher suite negotiated with the client. Both headers are in OpenSSL format. The possible values for the attribute are `true` and `false`. The default is `false`.

`routing.http.xff_client_port.enabled`

Indicates whether the `X-Forwarded-For` header should preserve the source port that the client used to connect to the load balancer. The possible values are `true` and `false`. The default is `false`.

`routing.http.xff_header_processing.mode`

Enables you to modify, preserve, or remove the `X-Forwarded-For` header in the HTTP request before the Application Load Balancer sends the request to the target. The possible values are `append`, `preserve`, and `remove`. The default is `append`.

- If the value is `append`, the Application Load Balancer adds the client IP address (of the last hop) to the `X-Forwarded-For` header in the HTTP request before it sends it to targets.
- If the value is `preserve`, the Application Load Balancer preserves the `X-Forwarded-For` header in the HTTP request, and sends it to targets without any change.

- If the value is `remove`, the Application Load Balancer removes the `X-Forwarded-For` header in the HTTP request before it sends it to targets.

`routing.http2.enabled`

Indicates whether clients can connect to the load balancer using HTTP/2. If `true`, clients can connect using HTTP/2 or HTTP/1.1. If `false`, clients must connect using HTTP/1.1. The default is `true`.

`waf.fail_open.enabled`

Indicates whether to allow a AWS WAF-enabled load balancer to route requests to targets if it is unable to forward the request to AWS WAF. The possible values are `true` and `false`. The default is `false`.

Note

The `routing.http.drop_invalid_header_fields.enabled` attribute was introduced to offer HTTP desync protection. The `routing.http.desync_mitigation_mode` attribute was added to provide more comprehensive protection from HTTP desync for your applications. You aren't required to use both attributes and can choose whichever attribute best meets your application's requirements.

IP address type

You can set the types of IP addresses that clients can use to access your internet-facing and internal load balancers.

Application Load Balancers support the following IP address types:

`ipv4`

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

`dualstack`

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

`dualstack-without-public-ipv4`

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

Considerations

- The load balancer communicates with targets based on the IP address type of the target group.
- When you enable dualstack mode for the load balancer, ELB provides an AAAA DNS record for the load balancer. Clients that communicate with the load balancer using IPv4 addresses resolve the A DNS record. Clients that communicate with the load balancer using IPv6 addresses resolve the AAAA DNS record.
- Access to your internal dualstack load balancers through the internet gateway is blocked to prevent unintended internet access. However, this does not prevent non-IGW internet access (such as, through peering, Transit Gateway, AWS Direct Connect, or Site-to-Site VPN).
- Application Load Balancer authentication only supports IPv4 when connecting to an Identity Provider (IdP) or Amazon Cognito endpoint. Without a public IPv4 address, the load balancer can't complete the authentication process, resulting in HTTP 500 errors.

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

Application Load Balancer IP Address Management

Application Load Balancers use Public Elastic IPv4 addresses from EC2's public IPv4 address pool. These IP addresses are visible in your AWS account when using the describe-addresses CLI, API or viewing the Elastic IPs (EIP) section in the AWS Console. Each ALB-associated IP address is marked with a service_managed attribute set to "ALB".

While these IPs are visible in your account, they remain fully managed by the Application Load Balancer service and cannot be modified or released. Application Load Balancer releases IPs back into the public IPv4 address pool when no longer in use.

CloudTrail logs API calls related to Application Load Balancer's EIP, such as the "AllocateAddress". These API calls are invoked by the Service Principal 'elasticloadbalancing.amazonaws.com'.

Note

Note: IPs allocated by Application Load Balancer do not count against your account's EIP limits.

IPAM IP address pools

An IPAM IP address pool is a collection of contiguous IP address ranges (or CIDRs) that you create using Amazon VPC IP Address Manager (IPAM). Using IPAM IP address pools with your Application Load Balancer enables you to organize your IPv4 addresses according to your routing and security needs. IPAM IP address pools give you the choice to bring some or all of your public IPv4 address ranges to AWS and use them with your Application Load Balancers. Your IPAM IP address pool is always prioritized when launching EC2 instances and creating Application Load Balancers. When your IP addresses are no longer in use, they become immediately available for use again.

To get started, create an IPAM IP address pool. For more information, see Bring your IP addresses to IPAM.

Considerations

- IPAM IPv6 address pools are not supported.
- IPAM IPv4 address pools are not supported with internal load balancers or the `dualstack-without-public-ipv4` IP address type.
- You can't delete an IP address in an IPAM IP address pool if it's currently in use by a load balancer.
- During the transition to a different IPAM IP address pool, existing connections are terminated according to the load balancers HTTP client keepalive duration.
- IPAM IP address pools can be shared across multiple accounts. For more information, see Configure integration options for your IPAM.
- There are no additional charges associated with using IPAM IP address pools with your load balancers. However, there might be charges related to IPAM, depending on which tier you use.

  If there are no more assignable IP addresses in your IPAM IP address pool, ELB uses AWS managed IPv4 addresses instead. There are additional charges to use AWS managed IPv4 addresses. To avoid these costs, you can add IP address ranges to your existing IPAM IP address pool.

For more information, see Amazon VPC pricing.

Load balancer connections

When processing a request, the load balancer maintains two connections: one connection with the client and one connection with a target. The connection between the load balancer and the client is also referred to as the front-end connection. The connection between the load balancer and the target is also referred to as the back-end connection.

Cross-zone load balancing

With Application Load Balancers, cross-zone load balancing is on by default and cannot be changed at the load balancer level. For more information, see the Cross-zone load balancing section in the *Elastic Load Balancing User Guide*.

Turning off cross-zone load balancing is possible at the target group level. For more information, see the section called "Turn off cross-zone load balancing".

DNS name

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

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 Application Load Balancer. When a client makes a request using this custom domain name, the DNS server resolves it to the DNS name for your Application 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 Application 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 Application Load Balancer. For more information, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.

The Application Load Balancer has one IP address per enabled Availability Zone. These are the IP addresses of the Application Load Balancer nodes. The DNS name of the Application Load

Balancer resolves to these addresses. For example, suppose that the custom domain name for your Application Load Balancer is `example.applicationloadbalancer.com`. Use the following **dig** or **nslookup** command to determine the IP addresses of the Application Load Balancer nodes.

Linux or Mac

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

Windows

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

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

Linux or Mac

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

Windows

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

Create an Application Load Balancer

An Application Load Balancer takes requests from clients and distributes them across targets in a target group, such as EC2 instances. For more information, see How ELB works. in the *Elastic Load Balancing User Guide*.

Tasks

- Prerequisites
- Create the load balancer
- Test the load balancer
- Next steps

Prerequisites

- Decide which Availability Zones and IP address types your application will support. Configure the 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. For more information, see the section called “Subnets for your load balancer”.
- 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 Recommended rules.
- 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 an Application 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 an Application Load Balancer

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose **Create load balancer**.
4. Under **Application Load Balancer**, choose **Create**.
5. **Basic configuration**

   a. For **Load balancer name**, enter a name for your load balancer. The name must be unique within your set of load balancers for the Region. Names can have a maximum of 32 characters, and can contain only alphanumeric characters and hyphens. They can not begin or end with a hyphen, or with `internal-`. You can't change the name of your Application Load Balancer after it's created.

b. For **Scheme**, choose **Internet-facing** or **Internal**. An internet-facing load balancer routes requests from clients to targets over the internet. An internal 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 load balancer or **Dualstack** if your clients use both IPv4 and IPv6 addresses to communicate with the load balancer. Choose **Dualstack without public IPv4** if your clients use only IPv6 addresses to communicate with the 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. (Optional) For **IP pools**, you can select **Use IPAM pool for public IPv4 addresses**. For more information, see the section called “IPAM IP address pools”.

c. For **Availability Zones and subnets**, enable zones for your load balancer as follows:

- Select subnets from at least two Availability Zones
- Select subnets from at least one Local Zone
- Select one Outpost subnet

For more information, see the section called “Subnets for your load balancer”.

With a **Dualstack** load balancer, you must select subnets with both IPv4 and IPv6 CIDR blocks.

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*.

8. **Listeners and routing**

a. The default is a listener that accepts HTTP 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, an HTTPS listener).

9. **Secure listener settings**

   This section appears only if you add an HTTPS listener.

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

   b. For **Default SSL/TLS certificate**, the following options are available:

      - If you created or imported a certificate using AWS Certificate Manager, choose **From ACM**, then choose the certificate.
      - If you imported a certificate using IAM, choose **From IAM**, and then choose your certificate.
      - 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) Select **Mutual authentication (mTLS)**, choose a policy to enable ALPN.

      For more information, see Mutual TLS authentication.

10. **Optimize with service integrations**

    (Optional) You can integrate other AWS with your load balancer. For more information, see Load balancer integrations.

11. **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.

12. **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 an Application 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 application \
    --subnets subnet-1234567890abcdef0 subnet-0abcdef1234567890 \
    --security-groups sg-1111222233334444
```

To create an internal Application Load Balancer

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

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

To create a dualstack Application Load Balancer

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

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --type application \
    --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 an HTTP listener and Create an HTTPS listener.

CloudFormation

To create an Application Load Balancer

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

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      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 an HTTP listener and Create an HTTPS listener.

Test the load balancer

After creating your load balancer, you can verify that your EC2 instances pass the initial health check. You can then check that the load balancer is sending traffic to your EC2 instance. To delete the load balancer, see Delete an Application Load Balancer.

To test the load balancer

1. After the load balancer is created, choose **Close**.

2. In the navigation pane, choose **Target Groups**.
3. Select the newly created target group.
4. Choose **Targets** and verify that your instances are ready. If the status of an instance is `initial`, it's typically because the instance is still in the process of being registered. This status can also indicate that the instance has not passed the minimum number of health checks to be considered healthy. After the status of at least one instance is healthy, you can test your load balancer. For more information, see Target health status.
5. In the navigation pane, choose **Load Balancers**.
6. Select the newly created load balancer.
7. Choose **Description** and copy the DNS name of the internet facing or internal load balancer (for example, my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.eu).

   - For internet facing load balancers, paste the DNS name into the address field of an internet connected web browser.
   - For internal load balancers, paste the DNS name into the address field of a web browser which has private connectivity to the VPC.

   If everything is configured correctly, the browser displays the default page of your server.
8. If the web page does not display, refer to the following documents for additional configuration help and troubleshooting steps.

   - For DNS related issues, see Routing traffic to an ELB load balancer in the *Amazon Route 53 Developer Guide*.
   - For Load Balancer related issues, see Troubleshoot your Application Load Balancers.

Next steps

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

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

Update the Availability Zones for your Application Load Balancer

You can enable or disable the Availability Zones for your load balancer at any time. After you enable an Availability Zone, the load balancer starts routing requests to the registered targets in that Availability Zone. Application Load Balancers have cross-zone load balancing on by default, resulting in requests being routed to all registered targets across all Availability Zones. When cross-zone load balancing is off, the load balancer only routes request to targets in the same Availability Zone. For more information, see Cross-zone load balancing. Your load balancer is most effective if you ensure that each enabled Availability Zone has at least one registered target.

After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer, but the load balancer will not route requests to them.

For more information, see the section called “Subnets for your load balancer”.

Console

To update Availability Zones

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 update Availability Zones

Use the set-subnets command.

```
aws elbv2 set-subnets \
    --load-balancer-arn load-balancer-arn \
    --subnets subnet-8360a9e7EXAMPLE subnet-b7d581c0EXAMPLE
```

CloudFormation

To update Availability Zones

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

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

Security groups for your Application Load Balancer

The security group for your Application Load Balancer controls the traffic that is allowed to reach and leave the load balancer. You must ensure that your load balancer can communicate with registered targets on both the listener port and the health check port. Whenever you add a listener to your load balancer or update the health check port for a target group used by the load balancer to route requests, you must verify that the security groups associated with the load balancer allow traffic on the new port in both directions. If they don't, you can edit the rules for the currently associated security groups or associate different security groups with the load balancer. You can choose the ports and protocols to allow. For example, you can open Internet Control Message Protocol (ICMP) connections for the load balancer to respond to ping requests (however, ping requests are not forwarded to any instances).

Considerations

- To ensure your targets receive traffic exclusively from the load balancer, restrict the security groups associated with your targets to accept traffic solely from the load balancer. This can be achieved by setting the load balancer's security group as the source in the ingress rule of the target's security group.
- If your Application Load Balancer is a target of an Network Load Balancer, the security groups for your Application Load Balancer use connection tracking to track information about traffic coming from the Network Load Balancer. This happens regardless of the security group rules set for your Application Load Balancer. For more information, see Security group connection tracking in the *Amazon EC2 User Guide*.
- We recommend that you allow inbound ICMP traffic to support Path MTU Discovery. For more information, see Path MTU Discovery in the *Amazon EC2 User Guide*.

Recommended rules

The following rules are recommended for an internet-facing load balancer with instances as targets.

Inbound		
Source	**Port Range**	**Comment**
0.0.0.0/0	*listener*	Allow all inbound traffic on the load balancer listener port
Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

The following rules are recommended for an internal load balancer with instances as targets.

Inbound		
Source	**Port Range**	**Comment**
VPC CIDR	*listener*	Allow inbound traffic from the VPC CIDR on the load balancer listener port
Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

The following rules are recommended for an Application Load Balancer with instances as targets that itself is a target of a Network Load Balancer.

Inbound		
Source	**Port Range**	**Comment**
client IP addresses/ CIDR	*alb listener*	Allow inbound client traffic on the load balancer listener port
VPC CIDR	*alb listener*	Allow inbound client traffic via AWS PrivateLink on the load balancer listener port

VPC CIDR	*alb listener*	Allow inbound health traffic from the Network Load Balancer
Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

Update the associated security groups

You can update the security groups associated with your load balancer at any time.

Console

To update security groups

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Security** tab, choose **Edit**.
5. To associate a security group with your load balancer, select it. To remove a security group association, choose the **X** icon for the security group.
6. Choose **Save changes**.

AWS CLI

To update security groups

Use the set-security-groups command.

```
aws elbv2 set-security-groups \
    --load-balancer-arn load-balancer-arn \
    --security-groups sg-01dd3383691d02f42 sg-00f4e409629f1a42d
```

CloudFormation

To update security groups

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
        - !Ref myNewSecurityGroup
```

Update the IP address types for your Application Load Balancer

You can configure your Application Load Balancer so that clients can communicate with the load balancer using IPv4 addresses only, or using both IPv4 and IPv6 addresses (dualstack). The 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 load balancer and update it at any time.
- The virtual private cloud (VPC) and subnets that you specify for the load balancer must have associated IPv6 CIDR blocks. For more information, see IPv6 addresses in the *Amazon EC2 User Guide*.
- The route tables for the load balancer subnets must route IPv6 traffic.
- The security groups for the load balancer must allow IPv6 traffic.

- The network ACLs for the load balancer subnets must allow IPv6 traffic.

Console

To update the IP address type

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Network mapping** tab, choose **Edit IP address type**.
5. For **IP address type**, choose **IPv4** to support IPv4 addresses only, **Dualstack** to support both IPv4 and IPv6 addresses, or **Dualstack without public IPv4** to support IPv6 addresses only.
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-alb
      Type: application
      Scheme: internal
```

```
      IpAddressType: dualstack
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
```

Update the IPAM IP address pools for your Application Load Balancer

IPAM IP address pools must first be created within IPAM before they can be used by your Application Load Balancer. For more information, see Bring your IP addresses to IPAM.

Console

To update the IPAM IP address pool

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Network mapping** tab, choose **Edit IP pools**.
5. Under **IP pools**, select **Use IPAM pool for public IPv4 addresses** and choose an IPAM pool.
6. Choose **Save changes**.

AWS CLI

To update the IPAM IP address pool

Use the modify-ip-pools command.

```
aws elbv2 modify-ip-pools \
    --load-balancer-arn load-balancer-arn \
    --ipam-pools Ipv4IpamPoolId=ipam-pool-1234567890abcdef0
```

CloudFormation

To update the IPAM IP address pool

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internet-facing
      IpAddressType: ipv4
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      Ipv4IpamPoolId: !Ref myIPAMPool
```

Edit attributes for your Application Load Balancer

After you create an Application Load Balancer, you can edit its attributes.

Load balancer attributes

- Connection idle timeout
- HTTP client keepalive duration
- Deletion protection
- Desync mitigation mode
- Host header preservation

Connection idle timeout

The connection idle timeout is the period of time an existing client or target connection can remain inactive, with no data being sent or received, before the load balancer closes the connection.

To ensure that lengthy operations such as file uploads have time to complete, send at least 1 byte of data before each idle timeout period elapses and increase the length of the idle timeout period as needed. We also recommend that you configure the idle timeout of your application to be larger than the idle timeout configured for the load balancer. Otherwise, if the application closes the

TCP connection to the load balancer ungracefully, the load balancer might send a request to the application before it receives the packet indicating that the connection is closed. If this is the case, then the load balancer sends an HTTP 502 Bad Gateway error to the client.

Application Load Balancers do not support HTTP/2 PING frames. These do not reset the connection idle timeout.

By default, ELB sets the idle timeout value for your load balancer to 60 seconds.

Console

To update the connection idle timeout value

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Traffic configuration**, enter a value for **Connection idle timeout**. The valid range is 1 through 4000 seconds.
6. Choose **Save changes**.

AWS CLI

To update the connection idle timeout value

Use the modify-load-balancer-attributes command with the `idle_timeout.timeout_seconds` attribute. The valid range is 1 to 4000 seconds.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=idle_timeout.timeout_seconds,Value=120"
```

CloudFormation

To update the connection idle timeout value

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `idle_timeout.timeout_seconds` attribute. The valid range is 1 to 4000 seconds.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "idle_timeout.timeout_seconds"
          Value: "120"
```

HTTP client keepalive duration

The HTTP client keepalive duration is the maximum length of time that an Application Load Balancer maintains a persistent HTTP connection to a client. After the configured HTTP client keepalive duration elapses, the Application Load Balancer accepts one more request and then returns a response that gracefully closes the connection.

The type of response sent by the load balancer depends on the HTTP version used by the client connection.

- For clients connected using HTTP 1.x, the load balancer sends an HTTP header containing the field `Connection: close`.
- For clients connected using HTTP/2, the load balancer sends a `GOAWAY` frame.

By default, Application Load Balancer sets the HTTP client keepalive duration value for load balancers to 3600 seconds, or 1 hour. The HTTP client keepalive duration cannot be turned off or set below the minimum of 60 seconds, but you can increase the HTTP client keepalive duration, up to a maximum of 604800 seconds, or 7 days. An Application Load Balancer begins the HTTP client keepalive duration period when an HTTP connection to a client is initially established. The duration period continues when there's no traffic, and does not reset until a new connection is established.

When load balancer traffic is shifted away from an impaired Availability Zone using zonal shift or zonal autoshift, clients with existing open connections might continue to make requests against

the impaired location until the clients reconnect. To support faster recovery, consider setting a lower keepalive duration value, to limit the amount of time that clients stay connected to a load balancer. For more information, see Limit the time that clients stay connected to your endpoints in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

> **Note**
>
> When the load balancer switches the IP address type of your Application Load Balancer to `dualstack-without-public-ipv4`, the load balancer waits for all active connections to complete. To decrease the amount of time it takes to switch the IP address type for your Application Load Balancer, consider lowering the HTTP client keepalive duration.

The Application Load Balancer assigns the HTTP client keepalive duration value during the initial connection. When you update the HTTP client keepalive duration, this can result in simultaneous connections with different HTTP client keepalive duration values. Existing connections retain the HTTP client keepalive duration value applied during its initial connection. New connections receive the updated HTTP client keepalive duration value.

Console

To update the client keepalive duration

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Traffic configuration**, enter a value for **HTTP client keepalive duration**. The valid range is 60 to 604800 seconds.
6. Choose **Save changes**.

AWS CLI

To update the client keepalive duration

Use the modify-load-balancer-attributes command with the `client_keep_alive.seconds` attribute. The valid range is 60 to 604800 seconds.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=client_keep_alive.seconds,Value=7200"
```

CloudFormation

To update the client keepalive duration

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `client_keep_alive.seconds` attribute. The valid range is 60 to 604800 seconds.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "client_keep_alive.seconds"
          Value: "7200"
```

Deletion protection

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

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

Console

To enable or disable deletion protection

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.

2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
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-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "deletion_protection.enabled"
          Value: "true"
```

Desync mitigation mode

Desync mitigation mode protects your application from issues due to HTTP desync. The load balancer classifies each request based on its threat level, allows safe requests, and then mitigates risk as specified by the mitigation mode that you specify. The desync mitigation modes are monitor, defensive, and strictest. The default is the defensive mode, which provides durable mitigation against HTTP desync while maintaining the availability of your application. You can switch to strictest mode to ensure that your application receives only requests that comply with RFC 7230.

The http_desync_guardian library analyzes HTTP requests to prevent HTTP desync attacks. For more information, see HTTP Desync Guardian on GitHub.

Classifications

The classifications are as follows:

- Compliant — Request complies with RFC 7230 and poses no known security threats.
- Acceptable — Request does not comply with RFC 7230 but poses no known security threats.
- Ambiguous — Request does not comply with RFC 7230 but poses a risk, as various web servers and proxies could handle it differently.
- Severe — Request poses a high security risk. The load balancer blocks the request, serves a 400 response to the client, and closes the client connection.

If a request does not comply with RFC 7230, the load balancer increments the `DesyncMitigationMode_NonCompliant_Request_Count` metric. For more information, see Application Load Balancer metrics.

The classification for each request is included in the load balancer access logs. If the request does not comply, the access logs include a classification reason code. For more information, see Classification reasons.

Modes

The following table describes how Application Load Balancers treat requests based on mode and classification.

Classification	Monitor mode	Defensive mode	Strictest mode
Compliant	Allowed	Allowed	Allowed
Acceptable	Allowed	Allowed	Blocked
Ambiguous	Allowed	Allowed[1]	Blocked
Severe	Allowed	Blocked	Blocked

[1] Routes the requests but closes the client and target connections. You might incur additional charges if your load balancer receives a large number of Ambiguous requests in Defensive mode. This is because the increased number of new connections per second contributes to the Load Balancer Capacity Units (LCU) used per hour. You can use the `NewConnectionCount` metric to compare how your load balancer establishes new connections in Monitor mode and Defensive mode.

Console

To update desync mitigation mode

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Traffic configuration**, **Packet handling**, for **Desync mitigation mode**, choose **Defensive**, **Strictest**, or **Monitor**.
6. Choose **Save changes**.

AWS CLI

To update desync mitigation mode

Use the modify-load-balancer-attributes command with the `routing.http.desync_mitigation_mode` attribute. The possible values are `monitor`, `defensive`, or `strictest`. The default is `defensive`.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=routing.http.desync_mitigation_mode,Value=monitor"
```

CloudFormation

To update desync mitigation mode

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `routing.http.desync_mitigation_mode` attribute. The possible values are `monitor`, `defensive`, or `strictest`. The default is `defensive`.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "routing.http.desync_mitigation_mode"
          Value: "monitor"
```

Host header preservation

When you enable the **Preserve host header** attribute, the Application Load Balancer preserves the `Host` header in the HTTP request, and sends the header to targets without any modification. If the Application Load Balancer receives multiple `Host` headers, it preserves all of them. Listener rules are applied only to the first `Host` header received.

By default, when the **Preserve host header** attribute is not enabled, the Application Load Balancer modifies the `Host` header in the following manner:

When host header preservation is not enabled, and listener port is a non-default port: When not using the default ports (ports 80 or 443) we append the port number to the host header if

it isn't already appended by the client. For example, the `Host` header in the HTTP request with `Host: www.example.com` would be modified to `Host: www.example.com:8080`, if the listener port is a non-default port such as `8080`.

When host header preservation is not enabled, and the listener port is a default port (port 80 or 443): For default listener ports (either port 80 or 443), we do not append the port number to the outgoing host header. Any port number that was already in the incoming host header, is removed.

The following table shows more examples of how Application Load Balancers treat host headers in the HTTP request based on listener port.

Listener port	Example request	Host header in the request	Host header preservation is disabled (default behavior)	Host header preservation is enabled
Request is sent on default HTTP/HTTPS listener.	`GET / index.html HTTP/1.1 Host: example.com`	example.com	example.com	example.com
Request is sent on default HTTP listener and host header has a port (for example, 80 or 443).	`GET / index.html HTTP/1.1 Host: example.com:80`	example.com:80	example.com	example.com:80
Request has an absolute path.	`GET https://dns_name/index.html HTTP/1.1 Host: example.com`	example.com	dns_name	example.com

Listener port	Example request	Host header in the request	Host header preservation is disabled (default behavior)	Host header preservation is enabled
Request is sent on a non-default listener port (for example, 8080)	`GET / index.html HTTP/1.1 Host: example.com`	example.com	example.com:8080	example.com
Request is sent on a non-default listener port and host header has port (for example, 8080).	`GET / index.html HTTP/1.1 Host: example.com:8080`	example.com:8080	example.com:8080	example.com:8080

Console

To enable host header preservation

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Attributes** tab, choose **Edit**.
5. Under **Packet handling**, turn on **Preserve host header**.
6. Choose **Save changes**.

AWS CLI

To enable host header preservation

Use the modify-load-balancer-attributes command with the `routing.http.preserve_host_header.enabled` attribute set to `true`.

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

CloudFormation

To enable host header preservation

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `routing.http.preserve_host_header.enabled` attribute.

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "routing.http.preserve_host_header.enabled"
          Value: "true"
```

Tag an Application Load Balancer

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

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

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

Restrictions

- Maximum number of tags per resource—50

- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

Console

To update the tags for a load balancer

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the 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.
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.

```
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.

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

CloudFormation

To add tags

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

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      Tags:
        - Key: 'project'
          Value: 'lima'
        - Key: 'department'
          Value: 'digital-media'
```

Delete an Application Load Balancer

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

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

Note that deleting a load balancer does not affect its registered targets. For example, your EC2 instances continue to run and are still registered to their target groups. To delete your target groups, see Delete an Application Load Balancer target group.

DNS records

If you have a DNS record for your domain that points to your load balancer, point it to a new location and wait for the DNS change to take effect before deleting your load balancer.

- 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.

Console

To delete a load balancer

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer, and then choose **Actions**, **Delete load balancer**.
4. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

AWS CLI

To delete a load balancer

Use the delete-load-balancer command.

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

View the Application Load Balancer resource map

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

To view the resource map for your Application Load Balancer

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. Choose the **Resource map** tab to display the load balancer's resource map.

Resource map components

Map views

There are two views available in the Application Load Balancer resource map: **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 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 groups

The Application Load Balancer resource map contains four resource groups, one for each resource type. The resource groups are **Listeners**, **Rules**, **Target groups**, and **Targets**.

Resource tiles

Each resource within a group 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.
  - **rule conditions:** The conditions for each rule.
  - **target group health summary:** The number of registered targets for each health status.
  - **target health status** The targets 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, `HTTP:80`
  - **Rules** - Select the rules action. For example, `Forward to target group`
  - **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 Application Load Balancer's resource map as a PDF.

Zonal shift for your Application Load Balancer

Zonal shift and zonal autoshift are features of Amazon Application Recovery Controller (ARC). With zonal shift, you can shift traffic 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.

With zonal autoshift, you authorize AWS to shift away resource traffic for an application from an Availability Zone during events, on your behalf, to help reduce time to recovery. AWS starts an autoshift when internal monitoring indicates that there is an Availability Zone impairment that could potentially impact customers. When AWS starts an autoshift, application traffic to resources that you've configured for zonal autoshift starts shifting away from the Availability Zone.

When you start a zonal shift, your load balancer stops sending new traffic for the resource to the affected Availability Zone. ARC creates the zonal shift immediately. However, it can take a short time for existing, in-progress connections in the Availability Zone to complete, depending on client behavior and connection reuse. Depending on your DNS settings and other factors, existing connections can complete in just a few minutes, or might take longer. For more information, see Limit the time that clients stay connected to your endpoints in the *Amazon Application Recovery Controller (ARC) Developer Guide*.

Contents

Before you begin a zonal shift

- Zonal shift is disabled by default and must be enabled on each Application Load Balancer. For more information, see Enable zonal shift for your Application Load Balancer.
- You can start a zonal shift for a specific load balancer only for a single Availability Zone. You can't start a zonal shift for multiple Availability Zones.
- AWS proactively removes zonal 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 your load balancers have cross-zone load balancing turned off and you use a zonal shift to remove a zonal load balancer IP address, the Availability Zone affected by the zonal shift also loses target capacity.

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

Cross-zone load balancing

When a zonal shift is started on an Application Load Balancer with cross-zone load balancing enabled, all traffic to targets is blocked in the availability zone being impacted, and zonal IP addresses are removed from DNS.

Benefits:

- Quicker recovery from availability zone failures.
- The ability to move traffic to a healthy availability zone if failures are detected in an availability zone.

- You can test application integrity by simulating and identifying failures to prevent unplanned downtime.

Zonal shift administrative override

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

When a zonal shift is started for a Application Load Balancer, all targets within the zone being shifted away from are considered administratively overridden. The Application 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

Enable zonal shift for your Application Load Balancer

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

Console

To enable zonal shift

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Select the Application 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-alb
      Type: application
      Scheme: internal
      IpAddressType: dualstack
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        -Key: "zonal_shift.config.enabled"
         Value: "true"
```

Start a zonal shift for your Application 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://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Select the Application 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 Application 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://eusc-de-east-1.console.amazonaws-eusc.eu/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 Application 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://eusc-de-east-1.console.amazonaws-eusc.eu/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, 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 Application Load Balancer

Load balancer Capacity Unit (LCU) reservations allow you to reserve a static minimum capacity for your load balancer. Application 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 ELB console, you can estimate how much capacity you need to reserve based on the reviewed traffic.

Alternatively, you can utilize the CloudWatch metric `PeakLCUs` to determine the level of capacity needed. The `PeakLCUs` metric accounts for peaks in your traffic pattern that the load balancer must scale across all scaling dimensions to support your workload. The `PeakLCUs` metric is different from the `ConsumedLCUs` metric, which only aggregates the billing dimensions of your traffic. Using the `PeakLCUs` metric is recommended to ensure your LCU reservation is adequate during load balancer scaling. When estimating capacity, use a per-minute `Sum` of `PeakLCUs`.

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.

Minimum and maximum values for an LCU reservation

The total reservation request must be at least 100 LCU. The maximum value is determined by the quotas for your account. For more information, see the section called “Load Balancer Capacity Units”.

Request Load balancer Capacity Unit reservation for your Application Load Balancer

Before you use LCU reservation, review the following:

- 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 a few minutes, 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://eusc-de-east-1.console.amazonaws-eusc.eu/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=100
```

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: 100
```

Update or cancel Load Balancer Capacity Unit reservations for your Application Load Balancer

If the traffic patterns for your load balancer change, you can update or cancel the LCU reservation for your load balancer. The status of the LCU reservation must be **Provisioned**.

Console

To update or cancel an LCU reservation

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 Application 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

The ReservedLCUs metric is reported on a per-minute basis. Capacity is reserved on an hourly basis. For example, if you have a LCU reservation of 6,000, the one-hour total for ReservedLCUs is 6,000, and the one-minute total is 100. To determine your reserved LCU utilization, refer to the PeakLCUs metric. You can set CloudWatch alarms to compare the per-minute Sum of PeakLCUs against your reserved capacity value, or the per-hour Sum of ReservedLCUs, to determine whether you have reserved enough capacity to meet your needs.

Console

To view the status of an LCU reservation

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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
```

Integrations for your Application Load Balancer

You can optimize your Application Load Balancer architecture by integrating with several other AWS services to enhance the performance, security, and availability of your application.

Load balancer integrations

- Amazon Application Recovery Controller (ARC)
- Amazon CloudFront + AWS WAF
- AWS Global Accelerator
- AWS Config
- AWS WAF

Amazon Application Recovery Controller (ARC)

Amazon Application Recovery Controller (ARC) helps you to shift traffic for your load balancer away from an impaired Availability Zone to a healthy Availability Zone in the same Region. Using zonal shift reduces the duration and severity that power outages, hardware issues, or software issues in an Availability Zone can have on your applications.

For more information, see Zonal shift for your Application Load Balancer.

Amazon CloudFront + AWS WAF

Amazon CloudFront is a web service that helps improve the performance, availability, and security of your applications that use AWS. CloudFront acts as a distributed, single point of entry for your web applications that use Application Load Balancers. It extends your Application Load Balancer's reach globally, allowing it to serve users efficiently from nearby edge locations, optimizing content delivery and reducing latency for users worldwide. The automatic content caching at these edge locations significantly reduces the load on your Application Load Balancer, improving its performance and scalability.

The one-click integration available in the ELB console creates a CloudFront distribution with the recommended AWS WAF security protections, and associates it to your Application Load Balancer. The AWS WAF protections block against common web exploits before reaching your load balancer. You can access the CloudFront distribution and its corresponding security dashboard from the load balancer's **Integrations** tab in the console. For more information, see Manage AWS WAF security protections in the CloudFront security dashboard in the *Amazon CloudFront Developer Guide* and Introducing CloudFront Security Dashboard, a Unified CDN and Security Experience at *aws.amazon.com/blogs*.

As a security best practice, configure your internet-facing Application Load Balancer's security groups to allow inbound traffic only from the AWS-managed prefix list for CloudFront, and remove any other inbound rules. For more information, see Use the CloudFront managed prefix list, Configure CloudFront to add a custom HTTP header to requests and Configure an Application Load Balancer to only forward requests that contain a specific header in the *Amazon CloudFront Developer Guide*>.

> **Note**
>
> CloudFront only supports ACM certificates in the US East (N. Virginia) us-east-1 region. If your Application Load Balancer has an HTTPS listener configured with an ACM certificate

in a region other than us-east-1, you will need to either change the CloudFront origin connection from HTTPS to HTTP, or provision an ACM certificate in the US East (N. Virginia) region and attach it to your CloudFront distribution.

AWS Global Accelerator

To optimize application availability, performance, and security, create an accelerator for your load balancer. The accelerator directs traffic over the AWS global network to static IP addresses that serve as fixed endpoints in the nearest Region to the client. AWS Global Accelerator is protected by Shield Standard, which minimizes application downtime and latency from DDoS attacks.

For more information, see Adding an accelerator when you create a load balancer in the *AWS Global Accelerator Developer Guide*.

AWS Config

To optimize monitoring and compliance of your load balancer, set up AWS Config. AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time. AWS Config streamlines audits, compliance, and troubleshooting.

For more information, see the AWS Config Developer Guide.

AWS WAF

You can use AWS WAF with your Application Load Balancer to allow or block requests based on the rules in a web access control list (web ACL).

By default, if the load balancer cannot get a response from AWS WAF, it returns an HTTP 500 error and does not forward the request. If you need your load balancer to forward requests to targets even if it is unable to contact AWS WAF, you can enable AWS WAF fail open.

Pre-defined web ACLs

When enabling AWS WAF integration you can choose to automatically create a new web ACL with pre-defined rules. The pre-defined web ACL includes three AWS managed rules which offer protections against the most common security threats.

- `AWSManagedRulesAmazonIpReputationList` - The Amazon IP reputation list rule group blocks IP addresses typically associated with bots or other threats. For more information, see Amazon IP reputation list managed rule group in the *AWS WAF Developer Guide*.
- `AWSManagedRulesCommonRuleSet` - The core rule set (CRS) rule group provides protection against exploitation of a wide range of vulnerabilities, including some of the high risk and commonly occurring vulnerabilities described in OWASP publications such as OWASP Top 10. For more information, see Core rule set (CRS) managed rule group in the *AWS WAF Developer Guide*.
- `AWSManagedRulesKnownBadInputsRuleSet` - The Known bad inputs rule group blocks request patterns that are known to be invalid and are associated with exploitation or discovery of vulnerabilities. For more information, see Known bad inputs managed rule group in the *AWS WAF Developer Guide*.

For more information, see Using web ACLs in AWS WAF in the *AWS WAF Developer Guide*.

Listeners for your Application 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 Application Load Balancer, you must add at least one listener. If your load balancer has no listeners, it can't receive traffic from clients. The rules that you define for your listeners determine 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**: HTTP, HTTPS
- **Ports**: 1-65535

You can use an HTTPS 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 HTTPS, you must deploy at least one SSL server certificate on the listener. For more information, see Create an HTTPS listener for your Application Load Balancer.

If you must ensure that the targets decrypt HTTPS traffic instead of the load balancer, you can create a Network Load Balancer with a TCP listener on port 443. With a TCP listener, the load balancer passes encrypted traffic through to the targets without decrypting it. For more information, see the User Guide for Network Load Balancers.

WebSockets

Application Load Balancers provide native support for WebSockets. You can upgrade an existing HTTP/1.1 connection into a WebSocket (`ws` or `wss`) connection by using an HTTP connection upgrade. When you upgrade, the TCP connection used for requests (to the load balancer as well as to the target) becomes a persistent WebSocket connection between the client and the target through the load balancer. You can use WebSockets with both HTTP and HTTPS listeners. The options that you choose for your listener apply to WebSocket connections as well as to HTTP traffic. Websockets are not supported for requests routed to target groups that have enabled target optimizer. For more information, see How the WebSocket Protocol Works in the *Amazon CloudFront Developer Guide*.

HTTP/2

Application Load Balancers provide native support for HTTP/2 with HTTPS listeners. You can send up to 128 requests in parallel using one HTTP/2 connection. You can use the protocol version to send the request to the targets using HTTP/2. For more information, see Protocol version. Because HTTP/2 uses front-end connections more efficiently, you might notice fewer connections between clients and the load balancer. You can't use the server-push feature of HTTP/2.

Mutual TLS authentication for Application Load Balancers supports HTTP/2 in both passthrough and verify modes. For more information, see Mutual authentication with TLS in Application Load Balancer.

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

Listener attributes

The following are the listener attributes for Application Load Balancers:

`routing.http.request.x_amzn_mtls_clientcert_serial_number.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert-Serial-Number** HTTP request header.

`routing.http.request.x_amzn_mtls_clientcert_issuer.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert-Issuer** HTTP request header.

`routing.http.request.x_amzn_mtls_clientcert_subject.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert-Subject** HTTP request header.

`routing.http.request.x_amzn_mtls_clientcert_validity.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert-Validity** HTTP request header.

`routing.http.request.x_amzn_mtls_clientcert_leaf.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert-Leaf** HTTP request header.

`routing.http.request.x_amzn_mtls_clientcert.header_name`

Enables you to modify the header name of the **X-Amzn-Mtls-Clientcert** HTTP request header.

`routing.http.request.x_amzn_tls_version.header_name`

Enables you to modify the header name of the **X-Amzn-Tls-Version** HTTP request header.

`routing.http.request.x_amzn_tls_cipher_suite.header_name`

Enables you to modify the header name of the **X-Amzn-Tls-Cipher-Suite** HTTP request header.

`routing.http.response.server.enabled`

Enables you to allow or remove the HTTP response server header.

`routing.http.response.strict_transport_security.header_value`

Informs browsers that the site should only be accessed using HTTPS, and that any future attempts to access it using HTTP should automatically be converted to HTTPS.

`routing.http.response.access_control_allow_origin.header_value`

Specifies which origins are allowed to access the server.

`routing.http.response.access_control_allow_methods.header_value`

Returns which HTTP methods are allowed when accessing the server from a different origin.

`routing.http.response.access_control_allow_headers.header_value`

Specifies which headers can be used during the request.

`routing.http.response.access_control_allow_credentials.header_value`

Indicates whether the browser should include credentials such as cookies or authentication when making requests.

`routing.http.response.access_control_expose_headers.header_value`

Returns which headers the browser can expose to the requesting client.

`routing.http.response.access_control_max_age.header_value`

Specifies how long the results of a preflight request can be cached, in seconds.

`routing.http.response.content_security_policy.header_value`

Specifies restrictions enforced by the browser to help minimize the risk of certain types of security threats.

`routing.http.response.x_content_type_options.header_value`

Indicates whether the MIME types advertised in the **Content-Type** headers should be followed and not be changed.

`routing.http.response.x_frame_options.header_value`

Indicates whether the browser is allowed to render a page in a **frame**, **iframe**, **embed** or **object**.

Default action

Every listener has a default action, also known as the default rule. The default rule can't be deleted and is always performed last. You can create additional rules. These rules consist of a priority, one or more actions, and one or more conditions. You can add or edit rules at any time. For more information, see Listener rules.

Create an HTTP listener for your Application Load Balancer

A listener 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.

The information on this page helps you create an HTTP listener for your load balancer. To add an HTTPS listener to your load balancer, see Create an HTTPS listener for your Application Load Balancer.

Prerequisites

- To add a forward action to the default listener rule, you must specify an available target group. For more information, see Create a target group for your Application Load Balancer.
- You can specify the same target group in multiple listeners, but these listeners must belong to the same load balancer. To use a target group with a load balancer, you must verify that it is not used by a listener for any other load balancer.

Add an HTTP 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.

To add another listener rule, see Listener rules.

Console

To add an HTTP listener

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, choose **Add listener**.
5. For **Protocol**, choose **HTTP**. Keep the default port or enter a different port.
6. For **Default action**, select one of the following routing actions and provide the required information:

- **Forward to target groups** – Choose a target group. To add another target group, choose **Add target group**, choose a target group, review the relative weights, and update the weights as needed. You must enable group-level stickiness if you enabled stickiness on any of the target groups.

  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.
- **Redirect to URL** – Enter the URL by entering each part separately on the **URI parts** tab, or by entering the full address on the **Full URL** tab. For **Status code**, select either temporary (HTTP 302) or permanent (HTTP 301) based on your needs.
- **Return fixed response** – Enter the **Response code** to return for dropped client requests. Optionally, you can specify the **Content type** and a **Response body**.

7. (Optional) To add tags, expand **Listener tags**. Choose **Add new tag** and enter the tag key and tag value.
8. Choose **Add listener**.

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 create an HTTP listener

Use the create-listener command. The following example creates an HTTP listener with a default rule that forwards traffic to the specified target group.

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

To create a forward action that distributes traffic between two target groups, use the following `--default-actions` option instead. When specifying multiple target groups, you must provide a weight for each target group.

```
    --default-actions '[{
          "Type":"forward",
          "ForwardConfig":{
            "TargetGroups":[
              {"TargetGroupArn":"target-group-1-arn","Weight":50},
              {"TargetGroupArn":"target-group-2-arn","Weight":50}
            ]
          }
        }]'
```

CloudFormation

To create an HTTP listener

Define a resource of type AWS::ElasticLoadBalancingV2::Listener. The following example creates an HTTP listener with a default rule that forwards traffic to the specified target group.

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

To create a forward action that distributes traffic between multiple target groups, use the `ForwardConfig` property. When specifying multiple target groups, you must provide a weight for each target group.

```
Resources:
  myHTTPlistener:
  Type: 'AWS::ElasticLoadBalancingV2::Listener'
  Properties:
    LoadBalancerArn: !Ref myLoadBalancer
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - Type: "forward"
        ForwardConfig:
```

```
        TargetGroups:
          - TargetGroupArn: !Ref TargetGroup1
            Weight: 50
          - TargetGroupArn: !Ref TargetGroup2
            Weight: 50
```

SSL certificates for your Application Load Balancer

When you create a secure listener for your Application Load Balancer, you must deploy at least one certificate on the load balancer. The load balancer requires X.509 certificates (SSL/TLS server certificates). 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 Request a public certificate in the *AWS Certificate Manager User Guide*.

We recommend that you create certificates for your load balancer using AWS Certificate Manager (ACM). ACM supports RSA certificates with 2048, 3072, and 4096-bit key lengths, and all ECDSA certificates. ACM integrates with ELB 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 SSL/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 about importing certificates into ACM, see Importing certificates in the *AWS Certificate Manager User Guide*. For

more information about uploading certificates to IAM, see Working with server certificates in the *IAM User Guide*.

Default certificate

When you create an HTTPS listener, you must specify exactly one certificate. This certificate is known as the *default certificate*. You can replace the default certificate after you create the HTTPS 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 an HTTPS listener, you can add certificates to the certificate list. If you created the listener using the AWS Management Console, we added the default certificate to the certificate list for you. Otherwise, the certificate list is empty. 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)
- Expiration (prefer not expired)
- Hashing algorithm (prefer SHA over MD5). If there are multiple SHA certificates, prefer the highest SHA number.
- 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 Application Load Balancer

ELB uses a Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL 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. By default, the first cipher on the server's list that matches any one of the client's ciphers is selected for the secure connection.

Considerations

- An HTTPS 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 HTTPS listener:
  - **Console** – The default security policy is `ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09`.
  - **Other methods** (for example, the AWS CLI, AWS CloudFormation, and the AWS CDK) – The default security policy is `ELBSecurityPolicy-2016-08`.
  - To view the TLS protocol version (log field position 5) and key exchange (log field position 13) for connection requests to your load balancer, enable connection logging and examine the corresponding log entries. For more information, see Connection logs.
  - Security policies with PQ in their names offer hybrid post-quantum key exchange. For compatibility, they support both classical and post-quantum ML-KEM key exchange algorithms. Clients must support the ML-KEM key exchange to use hybrid post-quantum TLS for key exchange. The hybrid post-quantum policies support SecP256r1MLKEM768, SecP384r1MLKEM1024 and X25519MLKEM768 algorithms. For more information, see Post-quantum Cryptography.
  - AWS recommends implementing the new post-quantum TLS (PQ-TLS) based security policy#`ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09` or `ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09`. This policy ensures backward compatibility by supporting clients capable of negotiating hybrid PQ-TLS, TLS 1.3 only, or TLS 1.2 only, thereby minimizing service disruption during the transition to post-quantum cryptography. You can progressively migrate to more restrictive security policies as your client applications develop the capability to negotiate PQ-TLS for key exchange operations.
- To meet compliance and security standards that require disabling certain TLS protocol versions, or to support legacy clients requiring deprecated ciphers, you can use one of the `ELBSecurityPolicy-TLS-` security policies. To view the TLS protocol version for requests to your Application Load Balancer, enable access logging for your load balancer and examine the corresponding access log entries. For more information, see Access logs.
- You can restrict which security policies are available to users across your AWS accounts and AWS Organizations by using the ELB 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.
- Application Load Balancers support TLS resumption using PSK (TLS 1.3) and session IDs/ session Tickets (TLS 1.2 and older). Resumptions are only supported in connections to the same Application Load Balancer IP address. The 0-RTT Data feature and early_data extension are not implemented.
- Application Load Balancers do not support custom security policies.
- Application Load Balancers support SSL renegotiation for target connections only.

Compatibility

- All secure listeners attached to the same load balancer must use compatible security policies. To migrate all secure listeners for a load balancer to security policies that are not compatible with the ones that are currently in use, remove all but one of the secure listeners, change the security policy of the secure listener, and then create additional secure listeners.
  - FIPS post-quantum TLS policies and FIPS policies - **Compatible**
  - Post-quantum TLS policies and FIPS or FIPS post-quantum TLS polices - **Compatible**
  - TLS polices (non-FIPS, non-post-quantum) and FIPS or FIPS post-quantum TLS policies - **Not Compatible**
  - TLS polices (non-FIPS, non-post-quantum) and post-quantum TLS policies - **Not Compatible**

Backend connections

- 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 any of your listeners are using:
  - **FIPS post-quantum TLS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09`
  - **FIPS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04`
  - **Post-quantum TLS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-PQ-2025-09`
  - **TLS 1.3 policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-2021-06`
  - **Other TLS policy** - Backend connections use `ELBSecurityPolicy-2016-08`

Security policies

Example describe-ssl-policies commands

You can describe the protocols and ciphers for a security policy, or find a policy that meets your needs, using the describe-ssl-policies AWS CLI command.

The following example describes the specified policy.

```
aws elbv2 describe-ssl-policies \
    --names "ELBSecurityPolicy-TLS13-1-2-Res-2021-06"
```

The following example lists policies with the specified string in the policy name.

```
aws elbv2 describe-ssl-policies \
    --query "SslPolicies[?contains(Name,'FIPS')].Name"
```

The following example lists policies that support the specified protocol.

```
aws elbv2 describe-ssl-policies \
    --query "SslPolicies[?contains(SslProtocols,'TLSv1.3')].Name"
```

The following example lists policies that support the specified cipher.

```
aws elbv2 describe-ssl-policies \
    --query "SslPolicies[?Ciphers[?contains(Name,'TLS_AES_128_GCM_SHA256')]].Name"
```

The following example lists policies that do not support the specified cipher.

```
aws elbv2 describe-ssl-policies \
    --query 'SslPolicies[?length(Ciphers[?starts_with(Name,`AES128-GCM-SHA256`)]) ==
 `0`].Name'
```

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
ELBSecurityPolicy-TLS13-1-3-PQ-2025-09	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-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Res-2021-06	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09	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-TLS13-1-0-PQ-2025-09	Yes	Yes	Yes	Yes
ELBSecurityPolicy-TLS-1-2-Ext-2018-06	No	Yes	No	No

Security policies	TLS 1.3	TLS 1.2	TLS 1.1	TLS 1.0
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

Ciphers by policy

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

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-3-2021-06 ELBSecurityPolicy-TLS13-1-3-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
ELBSecurityPolicy-TLS13-1-2-2021-06 ELBSecurityPolicy-TLS13-1-2-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384
ELBSecurityPolicy-TLS13-1-2-Res-2021-06	• TLS_AES_128_GCM_SHA256

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09	• TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384
ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

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

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-0-2021-06 ELBSecurityPolicy-TLS13-1-0-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

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

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

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

Security policy	Ciphers
ELBSecurityPolicy-2016-08	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • 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 **IANA** – TLS_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-3-2021-06 • ELBSecurityPolicy-TLS13-1-3-PQ-2025-09	1301

Cipher name	Security policies	Cipher suite
	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09	

Cipher name	Security policies	Cipher suite
OpenSSL – TLS_AES_256_GCM_SHA384 **IANA** – TLS_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-3-2021-06 • ELBSecurityPolicy-TLS13-1-3-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09	1302

Cipher name	Security policies	Cipher suite
OpenSSL – TLS_CHACHA20_POLY1305_SHA256 **IANA** – TLS_CHACHA20_POLY1305_SHA256	• ELBSecurityPolicy-TLS13-1-3-2021-06 • ELBSecurityPolicy-TLS13-1-3-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09	1303

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-GCM-SHA256 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c02b

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES128-GCM-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c02f

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-SHA256 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c023

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES128-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c027

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-SHA **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c009
OpenSSL – ECDHE-RSA-AES128-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c013

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c02c

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-2021-06 • ELBSecurityPolicy-TLS13-1-2-Res-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c030

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c024

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES256-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS13-1-2-2021-06 • ELBSecurityPolicy-TLS13-1-2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c028

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-SHA **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c00a
OpenSSL – ECDHE-RSA-AES256-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	c014

Cipher name	Security policies	Cipher suite
OpenSSL – AES128-GCM-SHA256 **IANA** – TLS_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	9c

Cipher name	Security policies	Cipher suite
OpenSSL – AES128-SHA256 **IANA** – TLS_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	3c

Cipher name	Security policies	Cipher suite
OpenSSL – AES128-SHA **IANA** – TLS_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	2f

Cipher name	Security policies	Cipher suite
OpenSSL – AES256-GCM-SHA384 **IANA** – TLS_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	9d

Cipher name	Security policies	Cipher suite
OpenSSL – AES256-SHA256 **IANA** – TLS_RSA_WITH_AES_256_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext1-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-2-2017-01 • ELBSecurityPolicy-TLS-1-1-2017-01 • ELBSecurityPolicy-2016-08	3d

Cipher name	Security policies	Cipher suite
OpenSSL – AES256-SHA **IANA** – TLS_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06 • ELBSecurityPolicy-TLS13-1-2-Ext2-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-2021-06 • ELBSecurityPolicy-TLS13-1-0-2021-06 • ELBSecurityPolicy-TLS13-1-0-PQ-2025-09 • ELBSecurityPolicy-TLS-1-2-Ext-2018-06 • ELBSecurityPolicy-TLS-1-1-2017-01 • 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

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-TLS13-1-3-FIPS-PQ-2025-09	Yes	No	No	No
ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04	Yes	Yes	No	No

Security policies	TLS 1.3	TLS 1.2	TLS 1.1	TLS 1.0
ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04	Yes	Yes	No	No
ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09	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
ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	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 ELBSecurityPolicy-TLS13-1-3-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384
ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256

Security policy	Ciphers
	• ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384
ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA • ECDHE-ECDSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256
ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA • ECDHE-ECDSA-AES256-SHA

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

Security policy	Ciphers
ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA • ECDHE-ECDSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

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-TLS13-1-3-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	

Cipher name	Security policies	Cipher suite
OpenSSL – TLS_AES_256_GCM_SHA384 **IANA** – TLS_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-3-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	1302

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-GCM-SHA256 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c02b

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES128-GCM-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c02f

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-SHA256 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c023

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES128-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c027

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES128-SHA **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c009
OpenSSL – ECDHE-RSA-AES128-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c013

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c02c

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Res-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c030

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c024

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES256-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c028

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-ECDSA-AES256-SHA **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c00a
OpenSSL – ECDHE-RSA-AES256-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	c014

Cipher name	Security policies	Cipher suite
OpenSSL – AES128-GCM-SHA256 **IANA** – TLS_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	9c
OpenSSL – AES128-SHA256 **IANA** – TLS_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	3c

Cipher name	Security policies	Cipher suite
OpenSSL – AES128-SHA **IANA** – TLS_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	2f
OpenSSL – AES256-GCM-SHA384 **IANA** – TLS_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	9d

Cipher name	Security policies	Cipher suite
OpenSSL – AES256-SHA256 **IANA** – TLS_RSA_WITH_AES_256_CBC_SHA256	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	3d
OpenSSL – AES256-SHA **IANA** – TLS_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-PQ-2025-09 • ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04 • ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09	35

FS supported 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 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384
ELBSecurityPolicy-FS-1-2-Res-2019-08	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384
ELBSecurityPolicy-FS-1-2-2019-08	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA • ECDHE-ECDSA-AES256-SHA
ELBSecurityPolicy-FS-1-1-2019-08	• ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256

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

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 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-FS-1-2-Res-2020-10 • ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c02b
OpenSSL – ECDHE-RSA-AES128-GCM-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	• ELBSecurityPolicy-FS-1-2-Res-2020-10 • ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c02f
OpenSSL – ECDHE-ECDSA-AES128-SHA256 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c023
OpenSSL – ECDHE-RSA-AES128-SHA256 **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	• ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c027
OpenSSL – ECDHE-ECDSA-AES128-SHA	• ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c009

Cipher name	Security policies	Cipher suite
IANA – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA		
OpenSSL – ECDHE-RSA-AES128-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	• ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c013
OpenSSL – ECDHE-ECDSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-FS-1-2-Res-2020-10 • ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c02c
OpenSSL – ECDHE-RSA-AES256-GCM-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	• ELBSecurityPolicy-FS-1-2-Res-2020-10 • ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c030
OpenSSL – ECDHE-ECDSA-AES256-SHA384 **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c024

Cipher name	Security policies	Cipher suite
OpenSSL – ECDHE-RSA-AES256-SHA384 **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	• ELBSecurityPolicy-FS-1-2-Res-2019-08 • ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c028
OpenSSL – ECDHE-ECDSA-AES256-SHA **IANA** – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c00a
OpenSSL – ECDHE-RSA-AES256-SHA **IANA** – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	• ELBSecurityPolicy-FS-1-2-2019-08 • ELBSecurityPolicy-FS-1-1-2019-08 • ELBSecurityPolicy-FS-2018-06	c014

Create an HTTPS listener for your Application Load Balancer

A listener 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.

To create an HTTPS listener, you must deploy at least one SSL server certificate on your load balancer. The load balancer uses a server certificate to terminate the front-end connection and then decrypt requests from clients before sending them to the targets. You must also specify a security policy, which is used to negotiate secure connections between clients and the load balancer.

If you need to pass encrypted traffic to targets without the load balancer decrypting it, you can create a Network Load Balancer or Classic Load Balancer with a TCP listener on port 443. With a TCP listener, the load balancer passes encrypted traffic through to the targets without decrypting it.

The information on this page helps you create an HTTPS listener for your load balancer. To add an HTTP listener to your load balancer, see Create an HTTP listener for your Application Load Balancer.

Prerequisites

- To add a forward action to the default listener rule, you must specify an available target group. For more information, see Create a target group for your Application Load Balancer.
- You can specify the same target group in multiple listeners, but these listeners must belong to the same load balancer. To use a target group with a load balancer, you must verify that it is not used by a listener for any other load balancer.
- Application Load Balancers do not support ED25519 keys.

Add an HTTPS listener

You configure a listener with a protocol and a port for connections from clients to the load balancer. For more information, see Listener configuration.

When you create a secure listener, you must specify a security policy and a certificate. To add certificates to the certificate list, see the section called “Add certificates to the certificate list”.

You must configure a default rule for the listener. You can add other listener rules after you create the listener. For more information, see Listener rules.

Console

To add an HTTPS listener

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, choose **Add listener**.
5. For **Protocol**, choose **HTTPS**. Keep the default port or enter a different port.
6. (Optional) For **Pre-routing action**, select one of the following actions:
   - **Authenticate user** – Choose an identity provider and provide the required information. For more information, see Authenticate users using an Application Load Balancer.

- **Validate token** – Enter the JWKS endpoint, issues, and any additional claims. For more information, see Verify JWTs using an Application Load Balancer.

7. For **Routing action**, select one of the following actions:

   - **Forward to target groups** – Choose a target group. To add another target group, choose **Add target group**, choose a target group, review the relative weights, and update the weights as needed. You must enable group-level stickiness if you enabled stickiness on any of the target groups.

     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.
   - **Redirect to URL** – Enter the URL by entering each part separately on the **URI parts** tab, or by entering the full address on the **Full URL** tab. For **Status code**, select either temporary (HTTP 302) or permanent (HTTP 301) based on your needs.
   - **Return fixed response** – Enter the **Response code** to return for dropped client requests. Optionally, you can specify the **Content type** and a **Response body**.
8. For **Security policy**, we select the recommended security policy. You can select a different security policy as needed.
9. For **Default SSL/TLS certificate**, choose the default certificate. We also add the default certificate to the SNI list. You can select a certificate using one of the following options:

   - **From ACM** – Choose a certificate from **Certificate (from ACM)**, which displays the certificates available from AWS Certificate Manager.
   - **From IAM** – Choose a certificate from **Certificate (from IAM)**, which displays the certificates that you imported to AWS Identity and Access Management.
   - **Import certificate** – Choose a destination for your certificate; 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.
10. (Optional) To enable mutual authentication, under **Client certificate handling**, enable **Mutual authentication (mTLS)**.

    The default mode is **passthrough**. If you select **Verify with trust store**:

- By default, connections with expired client certificates are rejected. To change this behavior expand **Advanced mTLS settings**, then under **Client certificate expiration** select **Allow expired client certificates**.
- For **Trust store**, choose an existing trust store, or choose **New trust store** and provide the required information.

11. (Optional) To add tags, expand **Listener tags**. Choose **Add new tag** and enter the tag key and tag value.
12. Choose **Add listener**.

AWS CLI

To create an HTTPS listener

Use the create-listener command. The following example creates an HTTPS listener with a default rule that forwards traffic to the specified target group.

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

CloudFormation

To create an HTTPS listener

Define a resource of type AWS::ElasticLoadBalancingV2::Listener. The following example creates an HTTPS listener with a default rule that forwards traffic to the specified target group.

```
Resources:
  myHTTPSListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTPS
      Port: 443
```

```
      DefaultActions:
        - Type: "forward"
          TargetGroupArn: !Ref myTargetGroup
      SslPolicy: ELBSecurityPolicy-TLS13-1-2-2021-06
      Certificates:
        - CertificateArn: certificate-arn
```

Update an HTTPS listener for your Application Load Balancer

After you create an HTTPS listener, you can replace the default certificate, update the certificate list, or replace the security policy.

Tasks

- Replace the default certificate
- Add certificates to the certificate list
- Remove certificates from the certificate list
- Update the security policy
- HTTP header modification

Replace the default certificate

You can replace the default certificate for your listener using the following procedure. For more information, see Default certificate.

Console

To replace the default certificate

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** 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.

```
Resources:
  myHTTPSListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTPS
      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. If you created the listener using the AWS Management Console, we added the default certificate to the

certificate list for you. Otherwise, the certificate list is empty. Adding the default certificate to the certificate list ensures that this certificate is used with the SNI protocol even if it is replaced as the default certificate. For more information, see SSL certificates for your Application Load Balancer.

Console

To add certificates to the certificate list

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** 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 a certificate 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"
```

Remove certificates from the certificate list

You can remove certificates from the certificate list for an HTTPS 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://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select 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 a certificate 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 an HTTPS listener, you can select the security policy that meets your needs. When a new security policy is added, you can update your HTTPS listener to use the new security policy. Application Load Balancers do not support custom security policies. For more information, see Security policies for your Application 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.

Compatibility

- All secure listeners attached to the same load balancer must use compatible security policies. To migrate all secure listeners for a load balancer to security policies that are not compatible with the ones that are currently in use, remove all but one of the secure listeners, change the security policy of the secure listener, and then create additional secure listeners.
  - FIPS post-quantum TLS policies and FIPS policies - **Compatible**
  - Post-quantum TLS policies and FIPS or FIPS post-quantum TLS polices - **Compatible**
  - TLS polices (non-FIPS, non-post-quantum) and FIPS or FIPS post-quantum TLS policies - **Not Compatible**
  - TLS polices (non-FIPS, non-post-quantum) and post-quantum TLS policies - **Not Compatible**

Console

To update the security policy

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Security** tab, choose **Edit secure listener settings**.
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:
  myHTTPSListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTPS
      Port: 443
      DefaultActions:
        - Type: "forward"
          TargetGroupArn: !Ref myTargetGroup
```

```
    SslPolicy: ELBSecurityPolicy-TLS13-1-2-2021-06
    Certificates:
      - CertificateArn: certificate-arn
```

HTTP header modification

HTTP header modification enables you to rename specific load balancer generated headers, insert specific response headers, and disable server response header. Application Load Balancers support header modification for both request and response headers.

For more information, see Enable HTTP header modification for your Application Load Balancer.

Listener rules for your Application Load Balancer

The listener rules for your Application Load Balancer determine how it routes requests to targets. When a listener receives a request, it evaluates the request against each rule in priority order, starting with the lowest numbered rule. Each rule includes conditions to be met and the actions to perform when the conditions for the rule are met. This flexible routing mechanism allows you to implement sophisticated traffic distribution patterns, support multiple applications or microservices behind a single load balancer, and customize request handling based on your application's specific requirements.

Rule basics

- Each rule consists of the following components: priority, actions, conditions, and optional transforms.
- Each rule action has a type and the information required to perform the action.
- Each rule condition has a type and the information required to evaluate the condition.
- Each rule transform has a regular expression to match and a replacement string.
- When you create a listener, you define actions for the default rule. The default rule can't have conditions or transforms. If none of the conditions for any other rules are met, then the action for the default rule is performed.
- Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. You can't change the priority of the default rule.
- Each rule must include exactly one of the following actions: `forward`, `redirect`, or `fixed-response`, and it must be the last action to be performed.

- Each rule other than the default rule can optionally include one of the following conditions: `host-header`, `http-request-method`, `path-pattern`, and `source-ip`. It can also optionally include one or both of the following conditions: `http-header` and `query-string`.
- Each rule other than the default rule can optionally include one host header rewrite transform and one URL rewrite transform.
- You can specify up to three comparison strings per condition and up to five per rule.

Contents

Action types for listener rules

Actions determine how a load balancer handles requests when the conditions for a listener rule are satisfied. Each rule must have at least one action that specifies how to handle the matching requests. Each rule action has a type and configuration information. Application Load Balancers support the following action types for listener rules.

Action types

`authenticate-cognito`

[HTTPS listeners] Use Amazon Cognito to authenticate users. For more information, see User authentication.

`authenticate-oidc`

[HTTPS listeners] Use an identity provider that is compliant with OpenID Connect (OIDC) to authenticate users. For more information, see User authentication.

`fixed-response`

Return a custom HTTP response. For more information, see Fixed-response actions.

`forward`

Forward requests to the specified target groups. For more information, see Forward actions.

`jwt-validation`

Validate JWT access tokens in client requests. For more information, see JWT verification.

`redirect`

Redirect requests from one URL to another. For more information, see Redirect actions.

Action basics

- Each rule must include exactly one of the following routing actions: `forward`, `redirect`, or `fixed-response`, and it must be the last action to be performed.
- An HTTPS listener can have a rule with a user authentication action and a routing action.
- When there are multiple actions, the action with the lowest priority is performed first.
- If the protocol version is gRPC or HTTP/2, the only supported actions are `forward` actions.

Fixed-response actions

A `fixed-response` action drops client requests and returns a custom HTTP response. You can use this action to return a 2XX, 4XX, or 5XX response code and an optional message.

When a `fixed-response` action is taken, the action and the URL of the redirect target are recorded in the access logs. For more information, see Access log entries. The count of successful `fixed-response` actions is reported in the `HTTP_Fixed_Response_Count` metric. For more information, see Application Load Balancer metrics.

Example Example fixed response action

You can specify an action when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following action sends a fixed response with the specified status code and message body.

```
[
  {
      "Type": "fixed-response",
      "FixedResponseConfig": {
          "StatusCode": "200",
```

```
            "ContentType": "text/plain",
            "MessageBody": "Hello world"
        }
    }
]
```

Forward actions

A `forward` action routes requests to its target group. Before you add a `forward` action, create the target group and add targets to it. For more information, see Create a target group.

Distribute traffic to multiple target groups

If you specify multiple target groups for a `forward` action, you must specify a weight for each target group. Each target group weight is a value from 0 to 999. Requests that match a listener rule with weighted target groups are distributed to these target groups based on their weights. For example, if you specify two target groups, each with a weight of 10, each target group receives half the requests. If you specify two target groups, one with a weight of 10 and the other with a weight of 20, the target group with a weight of 20 receives twice as many requests as the other target group.

If you configure a rule to distribute traffic between weighted target groups and one of the target groups is empty or has only unhealthy targets, the load balancer does not automatically fail over to a target group with healthy targets.

Sticky sessions and weighted target groups

By default, configuring a rule to distribute traffic between weighted target groups does not guarantee that sticky sessions are honored. To ensure that sticky sessions are honored, enable target group stickiness for the rule. When the load balancer first routes a request to a weighted target group, it generates a cookie named AWSALBTG that encodes information about the selected target group, encrypts the cookie, and includes the cookie in the response to the client. The client should include the cookie that it receives in subsequent requests to the load balancer. When the load balancer receives a request that matches a rule with target group stickiness enabled and contains the cookie, the request is routed to the target group specified in the cookie.

Application Load Balancers do not support cookie values that are URL encoded.

With CORS (cross-origin resource sharing) requests, some browsers require `SameSite=None; Secure` to enable stickiness. In this case, ELB generates a second cookie, AWSALBTGCORS, which

includes the same information as the original stickiness cookie plus this `SameSite` attribute. Clients receive both cookies.

Example forward action with one target group

You can specify an action when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following action forwards requests to the specified target group.

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067"
              }
          ]
      }
  }
]
```

Example forward action with weighted target groups

The following action forwards requests to the two specified target groups, based on the weight of each target group.

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-targets/73e2d6bc24d8a067",
                  "Weight": 10
              },
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-targets/09966783158cda59",
                  "Weight": 20
              }
```

```
            ]
        }
    }
]
```

Example forward action with stickiness enabled

If you have a forward action with multiple target groups and one or more of the target groups has sticky sessions enabled, you must enable target group stickiness.

The following action forwards requests to the two specified target groups, with target group stickiness enabled. Requests that do not contain the stickiness cookies are routed based on the weight of each target group.

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-targets/73e2d6bc24d8a067",
                  "Weight": 10
              },
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-targets/09966783158cda59",
                  "Weight": 20
              }
          ],
          "TargetGroupStickinessConfig": {
              "Enabled": true,
              "DurationSeconds": 1000
          }
      }
  }
]
```

Redirect actions

A `redirect` action redirects client requests from one URL to another. You can configure redirects as either temporary (HTTP 302) or permanent (HTTP 301) based on your needs.

A URI consists of the following components:

```
protocol://hostname:port/path?query
```

You must modify at least one of the following components to avoid a redirect loop: protocol, hostname, port, or path. Any components that you do not modify retain their original values.

protocol

The protocol (HTTP or HTTPS). You can redirect HTTP to HTTP, HTTP to HTTPS, and HTTPS to HTTPS. You cannot redirect HTTPS to HTTP.

hostname

The hostname. A hostname is not case-sensitive, can be up to 128 characters in length, and consists of alpha-numeric characters, wildcards (* and ?), and hyphens (-).

port

The port (1 to 65535).

path

The absolute path, starting with the leading "/". A path is case-sensitive, can be up to 128 characters in length, and consists of alpha-numeric characters, wildcards (* and ?), & (using &), and the following special characters: _-.$/~"'@:+.

query

The query parameters. The maximum length is 128 characters.

You can reuse URI components of the original URL in the target URL using the following reserved keywords:

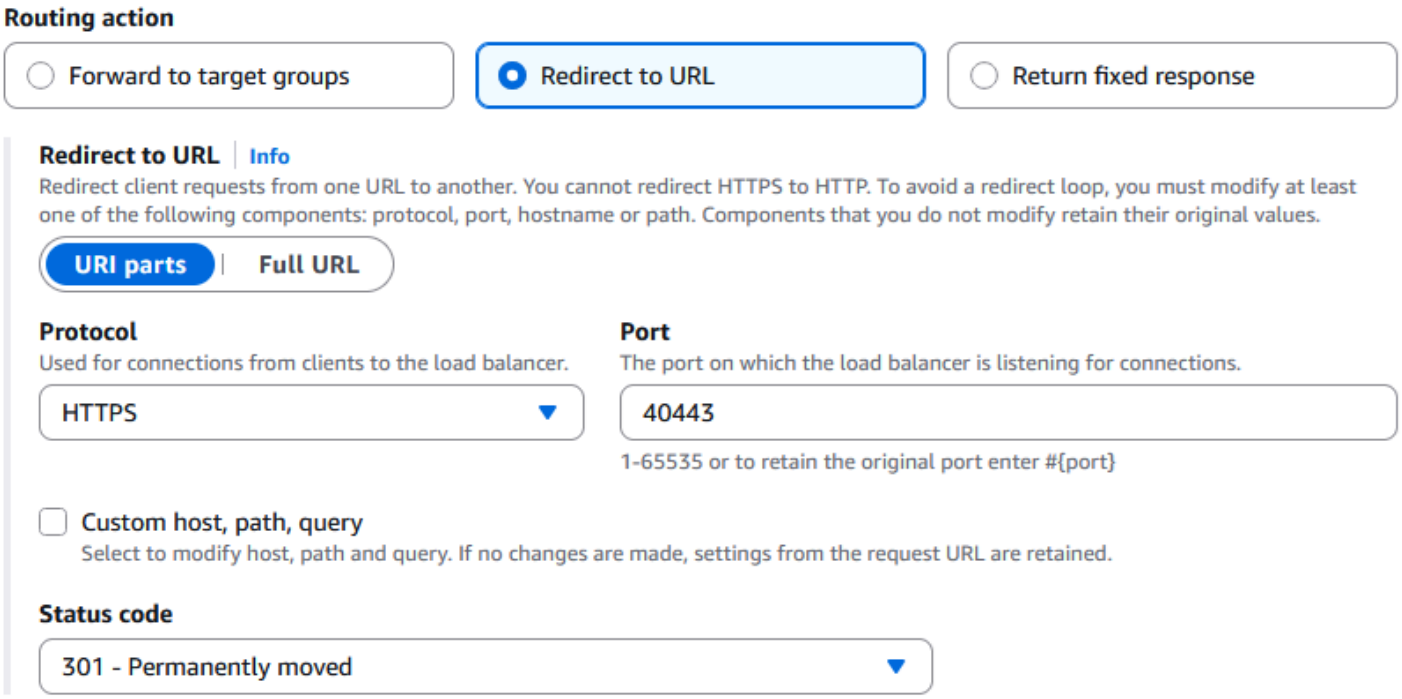
- `#{protocol}` - Retains the protocol. Use in the protocol and query components.
- `#{host}` - Retains the domain. Use in the hostname, path, and query components.
- `#{port}` - Retains the port. Use in the port, path, and query components.
- `#{path}` - Retains the path. Use in the path and query components.
- `#{query}` - Retains the query parameters. Use in the query component.

When a `redirect` action is taken, the action is recorded in the access logs. For more information, see Access log entries. The count of successful `redirect` actions is reported in the `HTTP_Redirect_Count` metric. For more information, see Application Load Balancer metrics.

Example redirect actions using the console

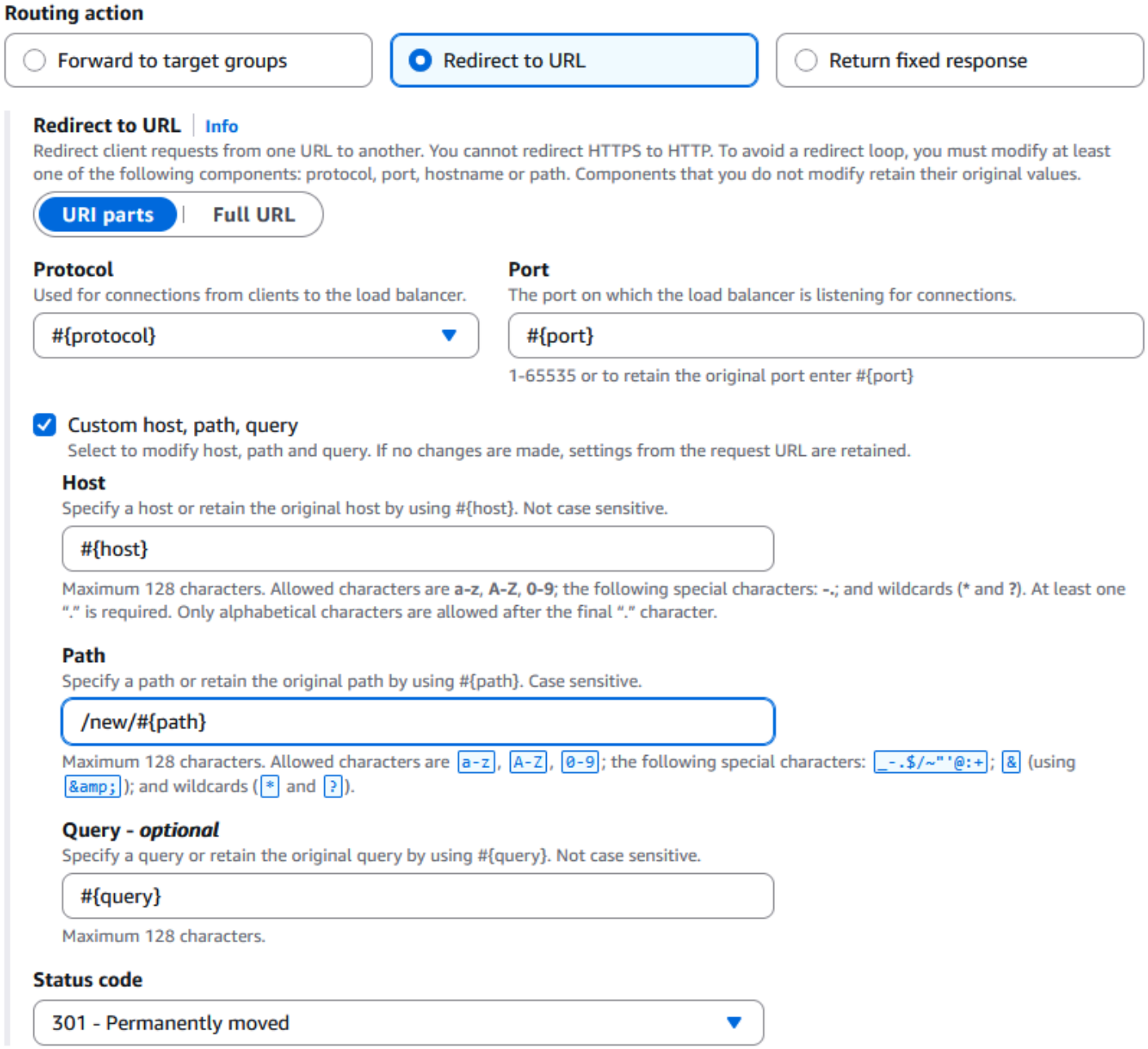
Redirect using HTTPS and port 40443

The following rule sets up a permanent redirect to a URL that uses the HTTPS protocol and the specified port (40443), but retains the original hostname, path, and query parameters. This screen is equivalent to "https://#{host}:40443/#{path}?#{query}".



Redirect using a modified path

The following rule sets up a permanent redirect to a URL that retains the original protocol, port, hostname, and query parameters, and uses the `#{path}` keyword to create a modified path. This screen is equivalent to "#{protocol}://#{host}:#{port}/new/#{path}?#{query}".



Example redirect actions using the AWS CLI

Redirect using HTTPS and port 40443

You can specify an action when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following action redirects an HTTP request to an HTTPS request on port 443, with the same host name, path, and query string as the HTTP request.

```
--actions '[{
    "Type": "redirect",
```

```
    "RedirectConfig": {
        "Protocol": "HTTPS",
        "Port": "443",
        "Host": "#{host}",
        "Path": "/#{path}",
        "Query": "#{query}",
        "StatusCode": "HTTP_301"
    }
}]'
```

Condition types for listener rules

Conditions define the criteria that incoming requests must meet for a listener rule to take effect. If a request matches the conditions for a rule, the request is handled as specified by the rule's actions. Each rule condition has a type and configuration information. Application Load Balancers support the following condition types for listener rules.

Condition types

`host-header`

Route based on the host name of each request. For more information, see Host conditions.

`http-header`

Route based on the HTTP headers for each request. For more information, see HTTP header conditions.

`http-request-method`

Route based on the HTTP request method of each request. For more information, see HTTP request method conditions.

`path-pattern`

Route based on path patterns in the request URLs. For more information, see Path conditions.

`query-string`

Route based on key/value pairs or values in the query strings. For more information, see Query string conditions.

`source-ip`

Route based on the source IP address of each request. For more information, see Source IP address conditions.

Condition basics

- Each rule can optionally include zero or one of each of the following conditions: `host-header`, `http-request-method`, `path-pattern`, and `source-ip`. Each rule can also include zero or more of each of the following conditions: `http-header` and `query-string`.
- With the `host-header`, `http-header`, and `path-pattern` conditions, you can use either value matching or regular expression (regex) matching.
- You can specify up to three match evaluations per condition. For example, for each `http-header` condition, you can specify up to three strings to be compared to the value of the HTTP header in the request. The condition is satisfied if one of the strings matches the value of the HTTP header. To require that all of the strings are a match, create one condition per match evaluation.
- You can specify up to five match evaluations per rule. For example, you can create a rule with five conditions where each condition has one match evaluation.
- You can include wildcard characters in the match evaluations for the `http-header`, `host-header`, `path-pattern`, and `query-string` conditions. There is a limit of five wildcard characters per rule.
- Rules are applied only to visible ASCII characters; control characters (0x00 to 0x1f and 0x7f) are excluded.

Demos

For demos, see Advanced request routing.

Host conditions

You can use host conditions to define rules that route requests based on the host name in the host header (also known as *host-based routing*). This enables you to support multiple subdomains and different top-level domains using a single load balancer.

A hostname is not case-sensitive, can be up to 128 characters in length, and can contain any of the following characters:

- A–Z, a–z, 0–9
- - .
- * (matches 0 or more characters)

- ? (matches exactly 1 character)

You must include at least one "." character. You can include only alphabetical characters after the final "." character.

Example hostnames

- example.com
- test.example.com
- *.example.com

The rule *.example.com matches test.example.com but doesn't match example.com.

Example Example host header condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands.

Value matching

```
[
  {
      "Field": "host-header",
      "HostHeaderConfig": {
          "Values": ["*.example.com"]
      }
  }
]
```

Regex matching

```
[
  {
      "Field": "host-header",
      "HostHeaderConfig": {
          "RegexValues": ["^(.*)\\.example\\.com$"]
      }
  }
]
```

HTTP header conditions

You can use HTTP header conditions to configure rules that route requests based on the HTTP headers for the request. You can specify the names of standard or custom HTTP header fields. The header name and the match evaluation are not case-sensitive. The following wildcard characters are supported in the comparison strings: * (matches 0 or more characters) and ? (matches exactly 1 character). Wildcard characters are not supported in the header name.

When the Application Load Balancer attribute `routing.http.drop_invalid_header_fields` is enabled, it will drop header names that don't conform to the regular expressions (`A-Z,a-z,0-9`). Header names that don't conform to the regular expressions can also be added.

Example Example HTTP header condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following condition is satisfied by requests with a User-Agent header that matches one of the specified strings.

Value matching

```
[
  {
      "Field": "http-header",
      "HttpHeaderConfig": {
          "HttpHeaderName": "User-Agent",
          "Values": ["*Chrome*", "*Safari*"]
      }
  }
]
```

Regex matching

```
[
  {
      "Field": "http-header",
      "HttpHeaderConfig": {
          "HttpHeaderName": "User-Agent",
          "RegexValues": [".+"]
      }
  }
]
```

HTTP request method conditions

You can use HTTP request method conditions to configure rules that route requests based on the HTTP request method of the request. You can specify standard or custom HTTP methods. The match evaluation is case-sensitive. Wildcard characters are not supported; therefore, the method name must be an exact match.

We recommend that you route GET and HEAD requests in the same way, because the response to a HEAD request may be cached.

Example Example HTTP method condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following condition is satisfied by requests that use the specified method.

```
[
  {
      "Field": "http-request-method",
      "HttpRequestMethodConfig": {
          "Values": ["CUSTOM-METHOD"]
      }
  }
]
```

Path conditions

You can use path conditions to define rules that route requests based on the URL in the request (also known as *path-based routing*).

The path pattern is applied only to the path of the URL, not to its query parameters. It is applied only to visible ASCII characters; control characters (0x00 to 0x1f and 0x7f) are excluded.

The rule evaluation is performed only after URI normalization occurs.

A path pattern is case-sensitive, can be up to 128 characters in length, and can contain any of the following characters.

- A–Z, a–z, 0–9
- _ - . $ / ~ " ' @ : +

- & (using &)
- * (matches 0 or more characters)
- ? (matches exactly 1 character)

If the protocol version is gRPC, conditions can be specific to a package, service, or method.

Example HTTP path patterns

- `/img/*`
- `/img/*/pics`

Example gRPC path patterns

- /package
- /package.service
- /package.service/method

The path pattern is used to route requests but does not alter them. For example, if a rule has a path pattern of `/img/*`, the rule forwards a request for `/img/picture.jpg` to the specified target group as a request for `/img/picture.jpg`.

Example Example path pattern condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following condition is satisfied by requests with a URL that contains the specified string.

Value matching

```
[
  {
      "Field": "path-pattern",
      "PathPatternConfig": {
          "Values": ["/img/*"]
      }
  }
]
```

Regex matching

```
[
  {
      "Field": "path-pattern",
      "PathPatternConfig": {
          "RegexValues": ["^\\/api\\/(.*)$"]
      }
  }
]
```

Query string conditions

You can use query string conditions to configure rules that route requests based on key/value pairs or values in the query string. The match evaluation is not case-sensitive. The following wildcard characters are supported: * (matches 0 or more characters) and ? (matches exactly 1 character).

Example Example query string condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following condition is satisfied by requests with a query string that includes either a key/value pair of "version=v1" or any key set to "example".

```
[
  {
      "Field": "query-string",
      "QueryStringConfig": {
          "Values": [
            {
                "Key": "version",
                "Value": "v1"
            },
            {
                "Value": "*example*"
            }
          ]
      }
  }
]
```

Source IP address conditions

You can use source IP address conditions to configure rules that route requests based on the source IP address of the request. The IP address must be specified in CIDR format. You can use both IPv4 and IPv6 addresses. Wildcard characters are not supported. You cannot specify the `255.255.255.255/32` CIDR for the source IP rule condition.

If a client is behind a proxy, this is the IP address of the proxy, not the IP address of the client.

This condition is not satisfied by the addresses in the X-Forwarded-For header. To search for addresses in the X-Forwarded-For header, use an `http-header` condition.

Example Example source IP condition

You can specify conditions when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following condition is satisfied by requests with a source IP address in one of the specified CIDR blocks.

```
[
  {
      "Field": "source-ip",
      "SourceIpConfig": {
          "Values": ["192.0.2.0/24", "198.51.100.10/32"]
      }
  }
]
```

Transforms for listener rules

A rule transform rewrites inbound requests before they are routed to targets. Rewriting a request does not change the routing decision made when evaluating the rule conditions. This is useful when clients send a different URL or host header than what the targets expect.

Using rule transforms offloads the responsibility for modifying paths, query strings, and host headers to the load balancer. This eliminates the need to add custom modification logic to your application code or rely on a third-party proxy to perform the modifications.

Application Load Balancers support the following transforms for listener rules.

Transforms

`host-header-rewrite`

Rewrites the host header in the request. The transform uses a regular expression to match a pattern in the host header and then replaces it with a replacement string.

`url-rewrite`

Rewrites the request URL. The transform uses a regular expression to match a pattern in the request URL and then replaces it with a replacement string.

Transform basics

- You can add one host header rewrite transform and one URL rewrite transform per rule.
- You can't add a transform to a default rule.
- If there is no pattern match, the original request is sent to the target.
- If there is a pattern match but the transform fails, we return an HTTP 500 error.

Host header rewrite transforms

You can modify the domain name specified in the host header.

Example Example host header transform

You can specify a transform when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following is an example host header transform. It transforms the host header to an internal endpoint.

```
[
  {
      "Type": "host-header-rewrite",
      "HostHeaderRewriteConfig": {
          "Rewrites": [
              {
                  "Regex": "^mywebsite-(.+).com$",
                  "Replace": "internal.dev.$1.myweb.com"
              }
          ]
```

```
      }
  }
]
```

For example, this transform rewrites the host header `https://mywebsite-example.com/project-a` as `https://internal.dev.example.myweb.com/project-a`.

URL rewrite transforms

You can modify the path or the query string of the URL. By rewriting the URL at the load balancer level, your frontend URLs can remain consistent for users and search engines even if your backend services change. You can also simplify complex URL query strings to make them easier for customers to type.

Note that you can't modify the protocol or port of the URL, only the path and the query string.

Example Example URL rewrite transform

You can specify a transform when you create or modify a rule. For more information, see the create-rule and modify-rule commands. The following is an example URL rewrite transform. It transforms the directory structure to a query string.

```
[
  {
      "Type": "url-rewrite",
      "UrlRewriteConfig": {
          "Rewrites": [
              {
                  "Regex": "^/dp/([A-Za-z0-9]+)/?$",
                  "Replace": "/product.php?id=$1"
              }
          ]
      }
  }
]
```

For example, this transform rewrites the request URL `https://www.example.com/dp/B09G3HRMW` as `https://www.example.com/product.php?id=B09G3HRMW`.

How URL rewrites differ from URL redirects

Characteristic	URL redirects	URL rewrites
URL display	Changes in the browser address bar	No change in the browser address bar
Status codes	Uses 301 (permanent) or 302 (temporary)	No status code change
Processing	Browser-side	Server-side
Common uses	Domain change, website consolida tion, fixing broken links	Clean URLs for SEO, hide complex structures, provide legacy URL mapping

Add a listener rule for your Application Load Balancer

You define a default rule when you create a listener. You can define additional rules at any time. Each rule must specify an action and a condition, and can optionally specify transforms. For more information, see the following:

- Action types
- Condition types
- Transforms

Console

To add a rule

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Rules** tab, choose **Add rule**.
6. (Optional) To specify a name for your rule, expand **Name and tags** and enter the name. To add additional tags, choose **Add additional tags** and enter the tag key and tag value.

7. For each condition, choose **Add condition**, choose the condition type, and provide the required condition values:

   - **Host header** – Select the match pattern type and enter the host header.

     Value matching – Maximum 128 characters. Not case sensitive. Allowed characters are a-z, A-Z, 0-9; the following special characters: -_.; and wildcards (* and ?). You must include at least one "." character. You can include only alphabetical characters after the final "." character.

     Regex matching – Maximum 128 characters.

   - **Path** – Select the match pattern type and enter the path.

     Value matching – Maximum 128 characters. Case sensitive. Allowed characters are a-z, A-Z, 0-9; the following special characters: _-.$/~"'@:+; &; and wildcards (* and ?).

     Regex matching – Maximum 128 characters.

   - **Query string** – Enter key:value pairs, or values without keys.

     Maximum 128 characters. Not case sensitive. Allowed characters are a-z, A-Z, 0-9; the following special characters: _-.$/~"'@:+&()!,;=; and wildcards (* and ?).

   - **HTTP request method** – Enter the HTTP request method.

     Maximum 40 characters. Case sensitive. Allowed characters are A-Z, and the following special characters: -_. Wildcards are not supported.

   - **HTTP header** – Select the match pattern type and enter the name of the header and the comparison strings.

     - **HTTP header name** – Rule will assess requests containing this header to confirm matching values.

       Value matching – Maximum 40 characters. Not case sensitive. Allowed characters are a-z, A-Z, 0-9, and the following special characters: *?-!#$%&'+.^_`|~. Wildcards are not supported.

       Regex matching – Maximum 128 characters.

     - **HTTP header value** – Enter strings to compare against the HTTP header value.

Value matching Maximum 128 characters. Not case sensitive. Allowed characters are a-z, A-Z, 0-9; spaces; the following special characters: !"#$%&'()+,./:;<=>@[]^_`{|}~-; and wildcards (* and ?).

Regex matching – Maximum 128 characters.

- **Source IP** – Define the source IP address in CIDR format. Both IPv4 and IPv6 CIDRs are allowed. Wildcards are not supported.

8. (Optional) To add a transform, choose **Add transform**, choose the transform type, and enter a regular expression to match and a replacement string.
9. (Optional, HTTPS listeners only) For **Pre-routing action**, select one of the following actions:
   - **Authenticate user** – Choose an identity provider and provide the required information. For more information, see Authenticate users using an Application Load Balancer.
   - **Validate token** – Enter the JWKS endpoint, issues, and any additional claims. For more information, see Verify JWTs using an Application Load Balancer.
10. For **Routing action**, select one of the following actions:
    - **Forward to target groups** – Choose a target group. To add another target group, choose **Add target group**, choose a target group, review the relative weights, and update the weights as needed. You must enable group-level stickiness if you enabled stickiness on any of the target groups.
    - **Redirect to URL** – Enter the URL by entering each part separately on the **URI parts** tab, or by entering the full address on the **Full URL** tab. For **Status code**, select either temporary (HTTP 302) or permanent (HTTP 301) based on your needs.
    - **Return fixed response** – Enter the **Response code** to return for dropped client requests. Optionally, you can specify the **Content type** and a **Response body**.
11. Choose **Next**.
12. For **Priority**, enter a value from 1-50,000. Rules are evaluated in priority order from the lowest value to the highest value.
13. Choose **Next**.
14. On the **Review and create** page, choose **Create**.

AWS CLI

To add a rule

Use the create-rule command.

The following example creates a rule with a `forward` action and a `host-header` condition.

```
aws elbv2 create-rule \
    --listener-arn listener-arn \
    --priority 10 \
    --conditions "Field=host-header,Values=example.com,www.example.com" \
    --actions "Type=forward,TargetGroupArn=target-group-arn"
```

To create a forward action that distributes traffic between two target groups, use the following `--actions` option instead.

```
    --actions '[{
        "Type":"forward",
        "ForwardConfig":{
          "TargetGroups":[
            {"TargetGroupArn":"target-group-1-arn","Weight":50},
            {"TargetGroupArn":"target-group-2-arn","Weight":50}
          ]
        }
    }]'
```

The following example creates a rule with a `fixed-response` action and a `source-ip` condition.

```
aws elbv2 create-rule \
    --listener-arn listener-arn \
    --priority 20 \
    --conditions '[{"Field":"source-ip","SourceIpConfig":{"Values":
["192.168.1.0/24","10.0.0.0/16"]}}]' \
    --actions "Type=fixed-
response,FixedResponseConfig={StatusCode=403,ContentType=text/
plain,MessageBody='Access denied'}"
```

The following example creates a rule with a `redirect` action and an `http-header` condition.

```
aws elbv2 create-rule \
    --listener-arn listener-arn \
    --priority 30 \
    --conditions '[{"Field":"http-header","HttpHeaderConfig":
{"HttpHeaderName":"User-Agent","Values":["*Mobile*","*Android*","*iPhone*"]}}]' \
```

```
    --actions
 "Type=redirect,RedirectConfig={Host=m.example.com,StatusCode=HTTP_302}"
```

CloudFormation

To add a rule

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

The following example creates a rule with a `forward` action and a `host-header` condition. The rule sends traffic to the specified target group when the condition is met.

```
Resources:
    myForwardListenerRule:
     Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
     Properties:
       ListenerArn: !Ref myListener
       Priority: 10
       Conditions:
         - Field: host-header
           Values:
             - example.com
             - www.example.com
       Actions:
         - Type: forward
           TargetGroupArn: !Ref myTargetGroup
```

Alternatively, to create a forward action that distributes traffic between two target groups when the condition is met, define `Actions` as follows.

```
       Actions:
         - Type: forward
           ForwardConfig:
             TargetGroups:
               - TargetGroupArn: !Ref TargetGroup1
                 Weight: 50
               - TargetGroupArn: !Ref TargetGroup2
                 Weight: 50
```

The following example creates a rule with a `fixed-response` action and a `source-ip` condition.

```
Resources:
    myFixedResponseListenerRule:
     Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
     Properties:
       ListenerArn: !Ref myListener
       Priority: 20
       Conditions:
         - Field: source-ip
           SourceIpConfig:
             Values:
                - 192.168.1.0/24
                - 10.0.0.0/16
       Actions:
         - Type: fixed-response
           FixedResponseConfig:
             StatusCode: 403
             ContentType: text/plain
             MessageBody: "Access denied"
```

The following example creates a rule with a `redirect` action and an `http-header` condition.

```
Resources:
    myRedirectListenerRule:
     Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
     Properties:
       ListenerArn: !Ref myListener
       Priority: 30
       Conditions:
         - Field: http-header
           HttpHeaderConfig:
             HttpHeaderName: User-Agent
             Values:
               - "*Mobile*"
               - "*Android*"
               - "*iPhone*"
       Actions:
         - Type: redirect
           RedirectConfig:
             Host: m.example.com
             StatusCode: HTTP_302
```

Edit a listener rule for your Application Load Balancer

You can edit the action and conditions for a listener rule at any time. Rule updates do not take effect immediately, so requests could be routed using the previous rule configuration for a short time after you update a rule. Any in-flight requests are completed.

Tasks

- Modify the default action
- Update rule priorities
- Update actions, conditions, and transforms
- Manage the rule tags

Modify the default action

The default action is assigned to a rule named **Default**. You can keep the current rule type and change the required information, or you can change the rule type and provide the new required information.

Console

To modify the default action

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Rules** tab, in the **Listener rules** section, select the default rule. Choose **Actions**, **Edit rule**.
6. Under **Default action**, update the actions as needed.

AWS CLI

To modify the default action

Use the modify-listener command. The following example updates the target group for the `forward` action.

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

The following example updates the default action to distribute traffic equally between two target groups.

```
aws elbv2 modify-listener \
    --listener-arn listener-arn \
    --default-actions '[{
      "Type":"forward",
      "ForwardConfig":{
        "TargetGroups":[
          {"TargetGroupArn":"target-group-1-arn","Weight":50},
          {"TargetGroupArn":"target-group-2-arn","Weight":50}
        ]
      }
    }]'
```

CloudFormation

To modify the default action

Update the AWS::ElasticLoadBalancingV2::Listener resource.

```
Resources:
    myHTTPlistener:
      Type: 'AWS::ElasticLoadBalancingV2::Listener'
      Properties:
        LoadBalancerArn: !Ref myLoadBalancer
        Protocol: HTTP
        Port: 80
        DefaultActions:
          - Type: "forward"
            TargetGroupArn: !Ref myNewTargetGroup
```

Update rule priorities

Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. You can change the priority of a nondefault rule at any time. You can't change the priority of the default rule.

Console

To update rule priorities

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Rules** tab, select the listener rule and then choose **Actions**, **Reprioritize rules**.
6. In the **Listener rules** section, the **Priority** column displays the current rule priorities. To update a rule priority, enter a value from 1-50,000.
7. Choose **Save changes**.

AWS CLI

To update rule priorities

Use the set-rule-priorities command.

```
aws elbv2 set-rule-priorities \
    --rule-priorities "RuleArn=listener-rule-arn,Priority=5"
```

CloudFormation

To update rule priorities

Update the AWS::ElasticLoadBalancingV2::ListenerRule resource.

```
Resources:
    myListenerRule:
     Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
     Properties:
       ListenerArn: !Ref myListener
       Priority: 5
       Conditions:
         - Field: host-header
           Values:
             - example.com
```

```
            - www.example.com
      Actions:
        - Type: forward
          TargetGroupArn: !Ref myTargetGroup
```

Update actions, conditions, and transforms

You can update the actions, conditions, and transforms for a rule.

Console

To update rule actions, conditions, and transforms

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Rules** tab, select the listener rule and then choose **Actions**, **Edit rule**.
6. Update the actions, conditions, and transforms as needed. For detailed steps, see Add a rule.
7. Choose **Next**.
8. (Optional) Update the priority.
9. Choose **Next**.
10. Choose **Save changes**.

AWS CLI

To update rule actions, conditions, and transforms

Use the modify-rule command. Include at least one of the following options: `--actions`, `--conditions`, and `--transforms`.

For examples of these options, see Add a rule.

CloudFormation

To update rule actions, conditions, and transforms

Update the AWS::ElasticLoadBalancingV2::ListenerRule resource.

For example rules, see Add a rule.

Manage the rule tags

Tags help you to categorize your listeners and rules in different ways. For example, you can tag a resource by purpose, owner, or environment. Tag keys must be unique for each rule. If you add a tag with a key that is already associated with the rule, it updates the value of that tag.

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

Console

To manage the tags for a rule

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. In the navigation pane, choose **Load Balancers**.
3. Choose the name of the load balancer to open its details page.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. On the **Rules** tab, select the text in the **Name tag** column to open the detail page for the rule.
6. On the rule details page, choose **Manage tags**.
7. On the **Manage tags** page, do one or more of the following:
   a. To add a tag, choose **Add new tag** and enter values for **Key** and **Value**.
   b. To delete a tag, choose **Remove** next to the tag.
   c. To update a tag, enter new values for **Key** or **Value**.
8. Choose **Save changes**.

AWS CLI

To add tags to a rule

Use the add-tags command.

```
aws elbv2 add-tags \
```

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

To remove tags from a rule

Use the remove-tags command.

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

CloudFormation

To add tags to a rule

Update the AWS::ElasticLoadBalancingV2::ListenerRule resource.

```
Resources:
    myListenerRule:
     Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
     Properties:
       ListenerArn: !Ref myListener
       Priority: 10
       Conditions:
         - Field: host-header
           Values:
             - example.com
             - www.example.com
       Actions:
         - Type: forward
           TargetGroupArn: !Ref myTargetGroup
       Tags:
        - Key: 'project'
          Value: 'lima'
        - Key: 'department'
          Value: 'digital-media'
```

Delete a listener rule for your Application Load Balancer

You can delete the nondefault rules for a listener at any time. You can't delete the default rule for a listener. When you delete a listener, all its rules are deleted.

Console

To delete a rule

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the text in the **Protocol:Port** column to open the detail page for the listener.
5. Select the rule.
6. Choose **Actions**, **Delete rule**.
7. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

AWS CLI

To delete a rule

Use the delete-rule command.

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

Mutual authentication with TLS in Application Load Balancer

Mutual TLS authentication is a variation of transport layer security (TLS). Traditional TLS establishes secure communications between a server and client, where the server needs to provide its identity to its clients. With mutual TLS, a load balancer negotiates mutual authentication between the client and the server while negotiating TLS. When you use mutual TLS with your Application Load Balancer, you simplify authentication management and reduce the load on your applications.

By using mutual TLS, your load balancer can manage client authentication to help ensure that only trusted clients communicate with your backend applications. When you use this feature, the load balancer authenticates clients using certificates from third-party certificate authority (CA) or by using the AWS Private Certificate Authority (PCA), optionally, with revocation checks. The load balancer passes the client certificate information to the backend using HTTP headers, which your applications can use for authorization.

Mutual TLS for Application Load Balancers provides the following options for validating your X.509v3 client certificates:

- **Mutual TLS passthrough:** The load balancer sends the entire client certificate chain to the target, without verifying it. Targets should verify the client certificate chain. Then, using the client certificate chain, you can implement the load balancer authentication and target authorization logic in your application.
- **Mutual TLS verify:** The load balancer performs X.509 client certificate authentication for clients when a load balancer negotiates TLS connections.

To use mutual TLS passthrough, you must configure the listener to accept the certificates from clients. To use mutual TLS with verification, see Configuring mutual TLS on an Application Load Balancer.

Before you begin configuring mutual TLS on your Application Load Balancer

Before you begin configuring mutual TLS on your Application Load Balancer, be aware of the following:

Quotas

Application Load Balancers include certain limits related to the amount of trust stores, CA certificates, and certificate revocation lists in use within your AWS account.

For more information, see Quotas for your Application Load Balancers.

Requirements for certificates

Application Load Balancers support the following for certificates used with mutual TLS authentication:

- Supported certificate: X.509v3
- Supported public keys: RSA 2K – 8K or ECDSA secp256r1, secp384r1, secp521r1
- Supported signature algorithms: SHA256, 384, 512 with RSA/SHA256, 384, 512 with EC/SHA256,384,512 hash with RSASSA-PSS with MGF1

CA certificate bundles

The following applies to certificate authority (CA) bundles:

- Application Load Balancers upload each certificate authority (CA) certificate bundle as a batch. Application Load Balancers don't support uploading individual certificates. If you need to add new certificates, you must upload the certificates bundle file.
- To replace a CA certificate bundle, use the ModifyTrustStore API.

Certificate order for passthrough

When you use mutual TLS passthrough, the Application Load Balancer inserts headers to present the clients certificate chain to the backend targets. The order of presentation starts with the leaf certificates and finishes with the root certificate.

Session resumption

Session resumption is not supported while using mutual TLS passthrough or verify modes with an Application Load Balancer.

HTTP headers

Application Load Balancers use `X-Amzn-Mtls` headers to send certificate information when it negotiates client connections using mutual TLS. For more information and example headers, see HTTP headers and mutual TLS.

CA certificate files

CA certificate files must satisfy the following requirements:

- Certificate file must use PEM (Privacy Enhanced Mail) format.
- Certificate contents must be enclosed within the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` boundaries.
- Comments must be preceded by a # character and must not contain any `-` characters.
- There cannot be any blank lines.

Example certificate that is not accepted (invalid):

```
# comments

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 01
    Signature Algorithm: ecdsa-with-SHA384
        Issuer: C=US, O=EXAMPLE, OU=EXAMPLE, CN=EXAMPLE
        Validity
```

```
            Not Before: Jan 11 23:57:57 2024 GMT
            Not After : Jan 10 00:57:57 2029 GMT
        Subject: C=US, O=EXAMPLE, OU=EXAMPLE, CN=EXAMPLE
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    00:01:02:03:04:05:06:07:08
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                00:01:02:03:04:05:06:07:08
            X509v3 Subject Alternative Name:
                URI:EXAMPLE.COM
    Signature Algorithm: ecdsa-with-SHA384
         00:01:02:03:04:05:06:07:08
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

Example certificates that are accepted (valid):

1. Single certificate (PEM–encoded):

```
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

2. Multiple certificates (PEM–encoded):

```
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

HTTP headers and mutual TLS

This section describes the HTTP headers that Application Load Balancers use to send certificate information when negotiating connections with clients using mutual TLS. The specific `X-Amzn-Mtls` headers that the Application Load Balancer uses depends on the mutual TLS mode that you've specified: passthrough mode or verify mode.

For information about other HTTP headers supported by Application Load Balancers, see HTTP headers and Application Load Balancers.

HTTP header for passthrough mode

For mutual TLS in passthrough mode, Application Load Balancers use the following header.

X-Amzn-Mtls-Clientcert

This header contains the URL-encoded PEM format of the entire client certificate chain presented in the connection, with `+=/` as safe characters.

Example header contents:

```
X-Amzn-Mtls-Clientcert: -----BEGIN%20CERTIFICATE-----%0AMIID<...reduced...>do0g
%3D%3D%0A-----END%20CERTIFICATE-----%0A-----BEGIN%20CERTIFICATE-----
%0AMIID1<...reduced...>3eZlyKA%3D%3D%0A-----END%20CERTIFICATE-----%0A
```

HTTP headers for verify mode

For mutual TLS in verify mode, Application Load Balancers use the following headers.

X-Amzn-Mtls-Clientcert-Serial-Number

This header contains a hexadecimal representation of the leaf certificate serial number.

Example header contents:

```
X-Amzn-Mtls-Clientcert-Serial-Number: 03A5B1
```

X-Amzn-Mtls-Clientcert-Issuer

This header contains an RFC2253 string representation of the issuer's distinguished name (DN).

Example header contents:

```
X-Amzn-Mtls-Clientcert-Issuer:
 CN=rootcamtls.com,OU=rootCA,O=mTLS,L=Seattle,ST=Washington,C=US
```

X-Amzn-Mtls-Clientcert-Subject

This header contains an RFC2253 string representation of the subject's distinguished name (DN).

Example header contents:

```
X-Amzn-Mtls-Clientcert-Subject: CN=client_.com,OU=client-3,O=mTLS,ST=Washington,C=US
```

X-Amzn-Mtls-Clientcert-Validity

This header contains an ISO8601 format of the `notBefore` and `notAfter` date.

Example header contents:

```
X-Amzn-Mtls-Clientcert-Validity:
 NotBefore=2023-09-21T01:50:17Z;NotAfter=2024-09-20T01:50:17Z
```

X-Amzn-Mtls-Clientcert-Leaf

This header contains a URL-encoded PEM format of the leaf certificate, with `+=/` as safe characters.

Example header contents:

```
X-Amzn-Mtls-Clientcert-Leaf: -----BEGIN%20CERTIFICATE-----%0AMIIG<...reduced...>NmrUlw
%0A-----END%20CERTIFICATE-----%0A
```

Advertise Certificate Authority (CA) subject name

Advertising Certificate Authority (CA) subject names enhances the authentication process by helping clients determine which certificates will be accepted during mutual TLS authentication.

When you enable Advertise CA subject names, the Application Load Balancer will advertise the list of Certificate Authorities (CAs) subject names that it trusts, based on the trust store it's associated

with. When a client connects to a target through the Application Load Balancer, the client receives the list of trusted CA subject names.

During the TLS handshake, when the Application Load Balancer requests a client certificate it includes a list of trusted CA Distinguished Names (DNs) in its Certificate Request message. This helps clients select valid certificates that match the advertised CA subject names, streamlining the authentication process and reducing connection errors.

You can enable Advertise CA subject name on new and existing listeners. For more information, see Add an HTTPS listener.

Connection logs for Application Load Balancers

ELB provides connection logs that capture attributes about the requests sent to your Application Load Balancers. Connection logs contain information such as the client IP address and port, client certificate information, connection results, and TLS ciphers being used. These connection logs can then be used to review request patterns, and other trends.

To learn more about connection logs, see Connection logs for your Application Load Balancer

Configuring mutual TLS on an Application Load Balancer

To use mutual TLS passthrough mode, you need only configure the listener to accept any certificates from clients. When you use mutual TLS passthrough, the Application Load Balancer sends the whole client certificate chain to the target using HTTP headers, which enables you to implement corresponding authentication and authorization logic in your application. For more information, see Create an HTTPS listener for your Application Load Balancer.

When you use mutual TLS in verify mode, the Application Load Balancer performs X.509 client certificate authentication for clients when a load balancer negotiates TLS connections.

To utilize mutual TLS verify mode, perform the following:

- Create a new trust store resource.
- Upload your certificate authority (CA) bundle and, optionally, revocation lists.
- Attach the trust store to the listener that is configured to verify client certificates.

Use the following procedures to configure mutual TLS verify mode on your Application Load Balancer.

Tasks

- Create a trust store
- Associate a trust store
- Replace a CA certificate bundle
- Add a certificate revocation list
- Delete a certificate revocation list
- Delete a trust store

Create a trust store

If you add a trust store when you create a load balancer or listener, the trust store is automatically associated with the new listener. Otherwise, you must associate it with a listener yourself.

Prerequisites

- To create a trust store, you must have a certificate bundle from your Certificate Authority (CA).

Console

The following example creates a trust store using the **Trust Store** portion of the console. Alternatively, you can create the trust store when you create an HTTP listener.

To create a trust store

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Trust Stores**.
3. Choose **Create trust store**.
4. **Trust store configuration**
   a. For **Trust store name**, enter a name for your trust store.
   b. For **Certificate authority bundle**, enter the Amazon S3 path to the ca certificate bundle to use.
   c. (Optional) Use **Object version** to select a previous version of the ca certificate bundle. Otherwise, the current version is used.
5. (Optional) For **Revocations**, you can add a certificate revocation list to your trust store.

a. Choose **Add new CRL** and enter the location of the certificate revocation list in Amazon S3.

b. (Optional) Use **Object version** to select a previous version of the certificate revocation list. Otherwise, the current version is used.

6. (Optional) Expand **Trust store tags** and enter up to 50 tags for your trust store.
7. Choose **Create trust store**.

AWS CLI

To create a trust store

Use the create-trust-store command.

```
aws elbv2 create-trust-store \
    --name my-trust-store \
    --ca-certificates-bundle-s3-bucket amzn-s3-demo-bucket \
    --ca-certificates-bundle-s3-key certificates/ca-bundle.pem
```

CloudFormation

To create a trust store

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

```
Resources:
  myTrustStore:
    Type: 'AWS::ElasticLoadBalancingV2::TrustStore'
    Properties:
      Name: my-trust-store
      CaCertificatesBundleS3Bucket: amzn-s3-demo-bucket
      CaCertificatesBundleS3Key: certificates/ca-bundle.pem
```

Associate a trust store

After you create a trust store, you must associate it with a listener before your Application Load Balancer can begin using the trust store. You can have only one trust store associated to each of your secure listeners, but one trust store can be associated to multiple listeners.

Console

You can associate a trust store with an existing listener, as shown in the following procedure. Alternatively, you can associate a trust store while creating an HTTPS listener. For more information, see Create an HTTPS listener.

To associate a trust store

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, choose the link in the **Protocol:Port** column to open the details page for the secure listener.
5. On the **Security** tab, choose **Edit secure listener settings**.
6. If mutual TLS is not enabled, select **Mutual authentication (mTLS)** under **Client certificate handling** and then choose **Verify with trust store**.
7. For **Trust store**, choose the trust store.
8. Choose **Save changes**.

AWS CLI

To associate a trust store

Use the modify-listener command.

```
aws elbv2 modify-listener \
    --listener-arn listener-arn \
    --mutual-authentication "Mode=verify,TrustStoreArn=trust-store-arn"
```

CloudFormation

To associate a trust store

Update the AWS::ElasticLoadBalancingV2::Listener resource.

```
Resources:
  myHTTPSListener:
```

```
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref myLoadBalancer
      Protocol: HTTPS
      Port: 443
      DefaultActions:
        - Type: "forward"
          TargetGroupArn: !Ref myTargetGroup
      SslPolicy: ELBSecurityPolicy-TLS13-1-2-2021-06
      Certificates:
        - CertificateArn: certificate-arn
      MutualAuthentication:
        - Mode: verify
          TrustStoreArn: trust-store-arn
```

Replace a CA certificate bundle

The CA certificate bundle is a required component of the trust store. It's a collection of trusted root and intermediate certificates that have been validated by a certificate authority. These validated certificates ensure the client can trust the certificate being presented is owned by the load balancer.

A trust store can only contain one CA certificate bundle at a time, but you can replace the CA certificate bundle at any time after the trust store is created.

Console

To replace a CA certificate bundle

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Trust Stores**.
3. Select the trust store.
4. Choose **Actions**, **Replace CA bundle**.
5. On the **Replace CA bundle** page, under **Certificate authority bundle**, enter the Amazon S3 location of the desired CA bundle.
6. (Optional) Use **Object version** to select a previous version of the certificate revocation list. Otherwise, the current version is used.
7. Select **Replace CA bundle**.

AWS CLI

To replace a CA certificate bundle

Use the modify-trust-store command.

```
aws elbv2 modify-trust-store \
    --trust-store-arn trust-store-arn \
    --ca-certificates-bundle-s3-bucket amzn-s3-demo-bucket-new \
    --ca-certificates-bundle-s3-key certificates/new-ca-bundle-pem
```

CloudFormation

To update the CA certificate bundle

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

```
Resources:
  myTrustStore:
    Type: 'AWS::ElasticLoadBalancingV2::TrustStore'
    Properties:
      Name: my-trust-store
      CaCertificatesBundleS3Bucket: amzn-s3-demo-bucket-new
      CaCertificatesBundleS3Key: certificates/new-ca-bundle.pem
```

Add a certificate revocation list

Optionally, you can create a certificate revocation list for a trust store. Revocation lists are released by certificate authorities and contain data for certificates that have been revoked. Application Load Balancers only support certificate revocation lists in the PEM format.

When a certificate revocation list is added to a trust store, it's given a revocation ID. The revocation IDs increase for every revocation list added to the trust store, and they can't be changed.

Application Load Balancers can't revoke certificates that have a negative serial number within a certificate revocation list.

Console

To add a revocation list

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.

2. On the navigation pane, choose **Trust Stores**.
3. Select the trust store to view it's details page.
4. On the **Certificate revocation lists** tab, select **Actions**, **Add revocation list**.
5. On the **Add revocation list** page, under **Certificate revocation list** enter the Amazon S3 location of the desired certificate revocation list
6. (Optional) Use **Object version** to select a previous version of the certificate revocation list. Otherwise the current version is used.
7. Select **Add revocation list**

AWS CLI

To add a revocation list

Use the add-trust-store-revocations command.

```
aws elbv2 add-trust-store-revocations \
    --trust-store-arn trust-store-arn \
    --revocation-contents "S3Bucket=amzn-s3-demo-bucket,S3Key=crl/revoked-
list.crl,RevocationType=CRL"
```

CloudFormation

To add a revocation list

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

```
Resources:
  myRevocationContents:
    Type: 'AWS:ElasticLoadBalancingV2::TrustStoreRevocation'
    Properties:
      TrustStoreArn: !Ref myTrustStore
      RevocationContents:
        - RevocationType: CRL
          S3Bucket: amzn-s3-demo-bucket
          S3Key: crl/revoked-list.crl
```

Delete a certificate revocation list

When you no longer need a certificate revocation list, you can delete it. When you delete a certificate revocation list from a trust store, it's revocation ID is also deleted and is not reused for the life of the trust store.

Console

To delete a revocation list

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Trust Stores**.
3. Select the trust store.
4. On the **Certificate revocation lists** tab, choose **Actions**, **Delete revocation list**.
5. When prompted for confirmation, enter **`confirm`**.
6. Choose **Delete**.

AWS CLI

To delete a revocation list

Use the remove-trust-store-revocations command.

```
aws elbv2 remove-trust-store-revocations \
    --trust-store-arn trust-store-arn \
    --revocation-ids id-1 id-2 id-3
```

Delete a trust store

When you no longer have use for a trust store, you can delete it. You can't delete a trust store that is associated with a listener.

Console

To delete a trust store

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Trust Stores**.

3. Select the trust store.
4. Choose **Delete**.
5. When prompted for confirmation, enter `confirm` and then choose **Delete**.

AWS CLI

To delete a trust store

Use the delete-trust-store command.

```
aws elbv2 delete-trust-store \
    --trust-store-arn trust-store-arn
```

Share your ELB trust store for Application Load Balancers

ELB integrates with AWS Resource Access Manager (AWS RAM) to enable trust store sharing. AWS RAM is a service that enables you to securely share your ELB trust store resources across AWS accounts and within your organization or organizational units (OUs). If you have multiple accounts, you can create a trust store once and use AWS RAM to make it usable by other accounts. If your account is managed by AWS Organizations, you can share trust stores with all the accounts in the organization or only accounts within specified organizational units (OUs).

With AWS RAM, you share resources that you own by creating a *resource share*. A resource share specifies the resources to share, and the consumers with whom to share them. In this model, the AWS account that owns the trust store (owner) shares it with other AWS accounts (consumers). Consumers can associate shared trust stores to their Application Load Balancer listeners in the same way they associate trust stores in their own account.

A trust store owner can share a trust store with:

- Specific AWS accounts inside or outside of its organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

Contents

- Permissions for shared trust stores
- Share a trust store
- Stop sharing a trust store
- Billing and metering

Prerequisites for trust store sharing

- You must create a resource share using AWS Resource Access Manager. For more information, see Create a resource share in the *AWS RAM User Guide*.
- To share a trust store, you must own it in your AWS account. You cannot share a trust store that has been shared with you.
- To share a trust store with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

Permissions for shared trust stores

Trust store owners

- Trust store owners can create a trust store.
- Trust store owners can use a trust store with load balancers in the same account.
- Trust store owners can share a trust store with other AWS accounts or AWS Organizations.
- Trust store owners can unshare a trust store from any AWS account or AWS Organizations.
- Trust store owners cannot prevent load balancers from using a trust store in the same account .
- Trust store owners can list all Application Load Balancers using a shared trust store.
- Trust store owners can delete a trust store if there are no current associations.
- Trust store owners can delete associations with a shared trust store.
- Trust store owners receive CloudTrail logs when a shared trust store is used.

Trust store consumers

- Trust store consumers can view shared trust stores.
- Trust store consumers can create or modify listeners using a trust store in the same account.
- Trust store consumers can create or modify listeners using a shared trust store.

- Trust store consumers cannot create a listener using a trust store that's no longer shared.
- Trust store consumers cannot modify a shared trust store.
- Trust store consumers can view a shared trust store ARN when associated to a listener.
- Trust store consumers receive CloudTrail logs when creating or modifying a listener using a shared trust store.

Managed permissions

When sharing a trust store, the resource share uses managed permissions to control which actions are allowed by the trust store consumer. You can use the default managed permissions `AWSRAMPermissionElasticLoadBalancingTrustStore`, which includes all available permissions, or create your own customer managed permissions. The `DescribeTrustStores`, `DescribeTrustStoreRevocations`, and `DescribeTrustStoreAssociations` permissions are always enabled and can not be removed.

The following permissions are supported for trust store resource shares:

elasticloadbalancing:CreateListener

Can attach a shared trust store to a new listener.

elasticloadbalancing:ModifyListener

Can attach a shared trust store to an existing listener.

elasticloadbalancing:GetTrustStoreCaCertificatesBundle

Can download the ca certificate bundle associated with the shared trust store.

elasticloadbalancing:GetTrustStoreRevocationContent

Can download the revocation file associated with the shared trust store.

elasticloadbalancing:DescribeTrustStores (Default)

Can list all trust stores owned and shared with the account.

elasticloadbalancing:DescribeTrustStoreRevocations (Default)

Can list all revocation content for the given trust store arn.

elasticloadbalancing:DescribeTrustStoreAssociations (Default)

Can list all resources in the trust store consumer account that are associated with the shared trust store.

Share a trust store

To share a trust store, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, the consumers with whom they are shared, and what actions principals can perform. When you share a trust store using the Amazon EC2 console, you add it to an existing resource share. To add the trust store to a new resource share, you must first create the resource share using the AWS RAM console.

When you share a trust store that you own with other AWS accounts, you enable those accounts to associate their Application Load Balancer listeners with trust stores in your account.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, consumers in your organization are automatically granted access to the shared trust store. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared trust store after accepting the invitation.

You can share a trust store that you own using the Amazon EC2 console, AWS RAM console, or the AWS CLI.

To share a trust store that you own using the Amazon EC2 console

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Trust Stores**.
3. Select the trust store name to view its details page.
4. On the **Sharing** tab, choose **Share trust store**.
5. On the **Share trust store** page, under **Resource shares**, select which resource shares your trust store will be shared with.
6. (Optional) If you need to create a new resource share, select the **Create a resource share in RAM console** link.
7. Select **Share trust store**.

To share a trust store that you own using the AWS RAM console

See Creating a Resource Share in the *AWS RAM User Guide*.

To share a trust store that you own using the AWS CLI

Use the create-resource-share command.

Stop sharing a trust store

To stop sharing a trust store that you own, you must remove it from the resource share. Existing associations persist after you stop sharing your trust store, however new associations to a previously shared trust store are not allowed. When either the trust store owner or the trust store consumer deletes an association, it is deleted from both accounts. If a trust store consumer wants to leave a resource share, they must ask the owner of the resource share to remove the account.

> ⚠ **Deleting associations**
>
> Trust store owners can forcefully delete existing trust store associations using the DeleteTrustStoreAssociation command. When an association is deleted, any load balancer listeners using the trust store can no longer verify client certificates and will fail TLS handshakes.

You can stop sharing a trust store using the Amazon EC2 console, AWS RAM console, or the AWS CLI.

To stop sharing a trust store that you own using the Amazon EC2 console

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Trust Stores**.
3. Select the trust store name to view its details page.
4. On the **Sharing** tab, under **Resource sharing**, select the resource shares to stop sharing with.
5. Choose **Remove**.

To stop sharing a trust store that you own using the AWS RAM console

See Updating a Resource Share in the *AWS RAM User Guide*.

To stop sharing a trust store that you own using the AWS CLI

Use the disassociate-resource-share command.

Billing and metering

Shared trust stores incur the same standard trust store rate, billed per hour, per trust store association with an Application Load Balancer.

For more information, including the specific rate per region, see ELB pricing

Authenticate users using an Application Load Balancer

You can configure an Application Load Balancer to securely authenticate users as they access your applications. This enables you to offload the work of authenticating users to your load balancer so that your applications can focus on their business logic.

The following use cases are supported:

- Authenticate users through an identity provider (IdP) that is OpenID Connect (OIDC) compliant.
- Authenticate users through social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito.
- Authenticate users through corporate identities, using SAML, OpenID Connect (OIDC), or OAuth, through the user pools supported by Amazon Cognito.

Prepare to use an OIDC-compliant IdP

Do the following if you are using an OIDC-compliant IdP with your Application Load Balancer:

- Create a new OIDC app in your IdP. The IdP's DNS must be publicly resolvable.
- You must configure a client ID and a client secret.
- Get the following endpoints published by the IdP: authorization, token, and user info. You can locate this information in the config.
- The IdP endpoints certificates should be issued by a trusted public certificate authority.
- The DNS entries for the endpoints must be publicly resolvable, even if they resolve to private IP addresses.
- Allow one of the following redirect URLs in your IdP app, whichever your users will use, where DNS is the domain name of your load balancer and CNAME is the DNS alias for your application:
  - https://*DNS*/oauth2/idpresponse
  - https://*CNAME*/oauth2/idpresponse

Prepare to use Amazon Cognito

Regions Available

Amazon Cognito integration for Application Load Balancers is available in the following regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)
- Canada (Central)
- Canada West (Calgary)
- Europe (Stockholm)
- Europe (Milan)
- Europe (Frankfurt)
- Europe (Zurich)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Spain)
- South America (São Paulo)
- Asia Pacific (Hong Kong)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Osaka)
- Asia Pacific (Mumbai)
- Asia Pacific (Hyderabad)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Jakarta)

- Asia Pacific (Melbourne)
- Middle East (UAE)
- Middle East (Bahrain)
- Africa (Cape Town)
- Israel (Tel Aviv)

Do the following if you are using Amazon Cognito user pools with your Application Load Balancer:

- Create a user pool. For more information, see Amazon Cognito user pools in the *Amazon Cognito Developer Guide*.
- Create a user pool client. You must configure the client to generate a client secret, use code grant flow, and support the same OAuth scopes that the load balancer uses. For more information, see Configuring a user pool app client in the *Amazon Cognito Developer Guide*.
- Create a user pool domain. For more information, see Configure a user pool domain in the *Amazon Cognito Developer Guide*.
- Verify that the requested scope returns an ID token. For example, the default scope, `openid` returns an ID token but the `aws.cognito.signin.user.admin` scope does not.
- To federate with a social or corporate IdP, enable the IdP in the federation section. For more information, see User pool sign-in with a third party identity provider in the *Amazon Cognito Developer Guide*.
- Allow the following redirect URLs in the callback URL field for Amazon Cognito, where DNS is the domain name of your load balancer, and CNAME is the DNS alias for your application (if you are using one):
  - https://*DNS*/oauth2/idpresponse
  - https://*CNAME*/oauth2/idpresponse
- Allow your user pool domain on your IdP app's callback URL. Use the format for your IdP. For example:
  - https://*domain-prefix*.auth.*region*.amazoncognito.com/saml2/idpresponse
  - https://*user-pool-domain*/saml2/idpresponse

The callback URL in the app client settings must use all lowercase letters.

To enable a user to configure a load balancer to use Amazon Cognito to authenticate users, you must grant the user permission to call the `cognito-idp:DescribeUserPoolClient` action.

Prepare to use Amazon CloudFront

Enable the following settings if you are using a CloudFront distribution in front of your Application Load Balancer:

- Forward request headers (all) — Ensures that CloudFront does not cache responses for authenticated requests. This prevents them from being served from the cache after the authentication session expires. Alternatively, to reduce this risk while caching is enabled, owners of a CloudFront distribution can set the time-to-live (TTL) value to expire before the authentication cookie expires.
- Query string forwarding and caching (all) — Ensures that the load balancer has access to the query string parameters required to authenticate the user with the IdP.
- Cookie forwarding (all) — Ensures that CloudFront forwards all authentication cookies to the load balancer.
- When configuring OpenID Connect (OIDC) authentication in conjunction with Amazon CloudFront, ensure that HTTPS port 443 is consistently used throughout the entire connection path. Otherwise, authentication failures can occur because the client OIDC redirect URLs do not match the port number of the originally generated URI.

Configure user authentication

You configure user authentication by creating an authenticate action for one or more listener rules. The `authenticate-cognito` and `authenticate-oidc` action types are supported only with HTTPS listeners. For descriptions of the corresponding fields, see AuthenticateCognitoActionConfig and AuthenticateOidcActionConfig in the *Elastic Load Balancing API Reference version 2015-12-01*.

The load balancer sends a session cookie to the client to maintain authentication status. This cookie always contains the `secure` attribute, because user authentication requires an HTTPS listener. This cookie contains the `SameSite=None` attribute with CORS (cross-origin resource sharing) requests.

For a load balancer supporting multiple applications that require independent client authentication, each listener rule with an authenticate action should have a unique cookie name. This ensures that clients are always authenticated with the IdP before being routed to the target group specified in the rule.

Application Load Balancers do not support cookie values that are URL encoded.

By default, the `SessionTimeout` field is set to 7 days. If you want shorter sessions, you can configure a session timeout as short as 1 second. For more information, see Session timeout.

Set the `OnUnauthenticatedRequest` field as appropriate for your application. For example:

- **Applications that require the user to log in using a social or corporate identity**—This is supported by the default option, `authenticate`. If the user is not logged in, the load balancer redirects the request to the IdP authorization endpoint and the IdP prompts the user to log in using its user interface.
- **Applications that provide a personalized view to a user that is logged in or a general view to a user that is not logged in**—To support this type of application, use the `allow` option. If the user is logged in, the load balancer provides the user claims and the application can provide a personalized view. If the user is not logged in, the load balancer forwards the request without the user claims and the application can provide the general view.
- **Single-page applications with JavaScript that loads every few seconds**—If you use the `deny` option, the load balancer returns an HTTP 401 Unauthorized error to AJAX calls that have no authentication information. But if the user has expired authentication information, it redirects the client to the IdP authorization endpoint.

The load balancer must be able to communicate with the IdP token endpoint (`TokenEndpoint`) and the IdP user info endpoint (`UserInfoEndpoint`). Application Load Balancers only support IPv4 when communicating with these endpoints. If your IdP uses public addresses, ensure the security groups for your load balancer and the network ACLs for your VPC allow access to the endpoints. When using an internal load balancer or the **IP address type** `dualstack-without-public-ipv4`, a NAT gateway can enable the load balancer to communicate with the endpoints. For more information, see NAT gateway basics in the *Amazon VPC User Guide*.

Use the following create-rule command to configure user authentication.

```
aws elbv2 create-rule \
    --listener-arn listener-arn \
    --priority 10 \
    --conditions Field=path-pattern,Values="/login" \
    --actions file://actions.json
```

The following is an example of the `actions.json` file that specifies an `authenticate-oidc` action and a `forward` action. `AuthenticationRequestExtraParams` allows you to pass extra

parameters to an IdP during authentication. Please follow documentation provided by your identity provider to determine the fields that are supported

```
[{
    "Type": "authenticate-oidc",
    "AuthenticateOidcConfig": {
        "Issuer": "https://idp-issuer.com",
        "AuthorizationEndpoint": "https://authorization-endpoint.com",
        "TokenEndpoint": "https://token-endpoint.com",
        "UserInfoEndpoint": "https://user-info-endpoint.com",
        "ClientId": "abcdefghijklmnopqrstuvwxyz123456789",
        "ClientSecret": "12345678901234567890123456789O",
        "SessionCookieName": "my-cookie",
        "SessionTimeout": 3600,
        "Scope": "email",
        "AuthenticationRequestExtraParams": {
            "display": "page",
            "prompt": "login"
        },
        "OnUnauthenticatedRequest": "deny"
    },
    "Order": 1
},
{
    "Type": "forward",
    "TargetGroupArn": "arn:aws:elasticloadbalancing:region-code:account-
id:targetgroup/target-group-name/target-group-id",
    "Order": 2
}]
```

The following is an example of the `actions.json` file that specifies an `authenticate-cognito` action and a `forward` action.

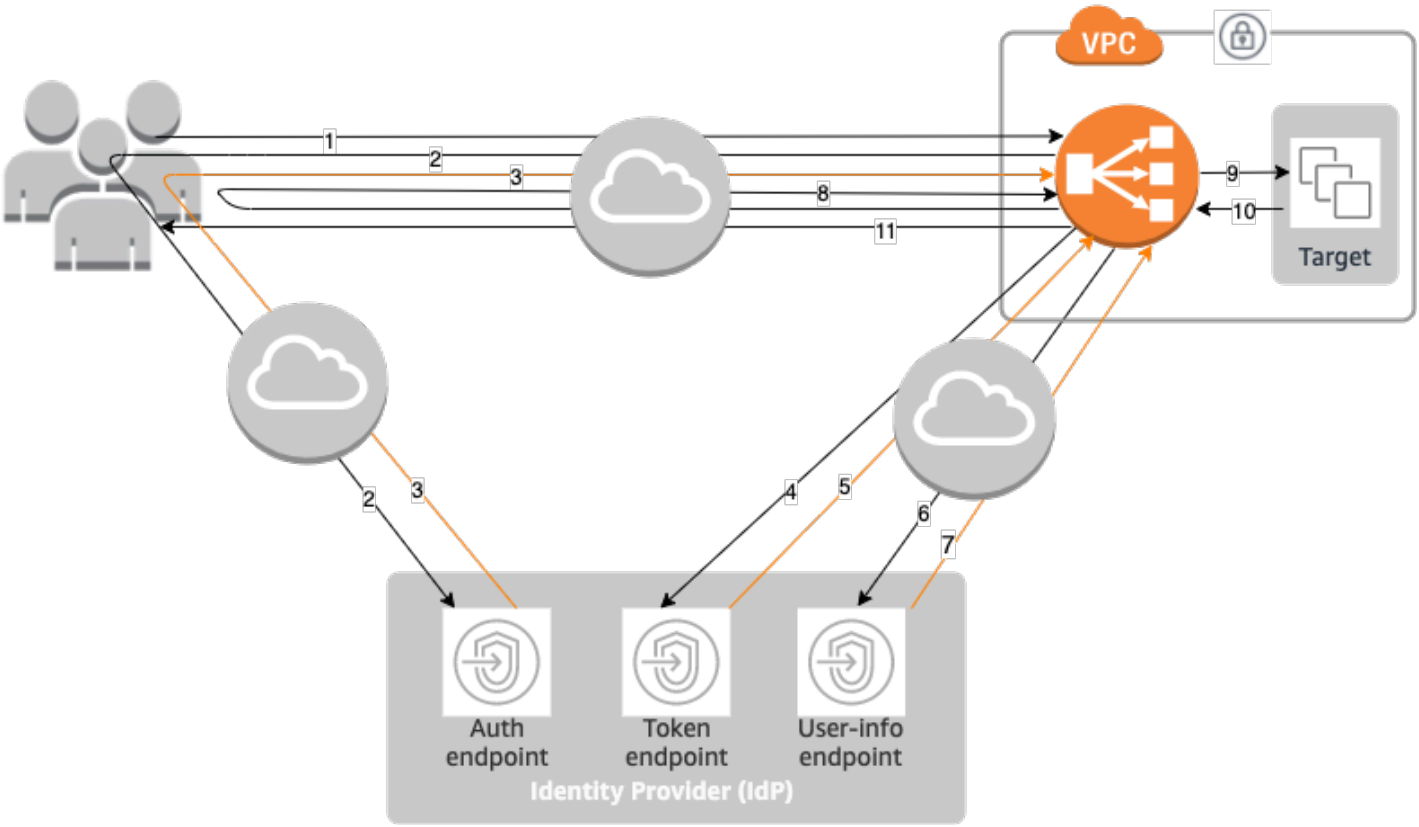
```
[{
    "Type": "authenticate-cognito",
    "AuthenticateCognitoConfig": {
        "UserPoolArn": "arn:aws:cognito-idp:region-code:account-id:userpool/user-pool-
id",
        "UserPoolClientId": "abcdefghijklmnopqrstuvwxyz123456789",
        "UserPoolDomain": "userPoolDomain1",
        "SessionCookieName": "my-cookie",
        "SessionTimeout": 3600,
        "Scope": "email",
```

```
        "AuthenticationRequestExtraParams": {
            "display": "page",
            "prompt": "login"
        },
        "OnUnauthenticatedRequest": "deny"
    },
    "Order": 1
},
{
    "Type": "forward",
    "TargetGroupArn": "arn:aws:elasticloadbalancing:region-code:account-
id:targetgroup/target-group-name/target-group-id",
    "Order": 2
}]
```

For more information, see Listener rules for your Application Load Balancer.

Authentication flow

The following network diagram is a visual representation of how an Application Load Balancer uses OIDC to authenticate users.



The numbered items below, highlight and explain elements shown in the preceding network diagram.

1. User sends an HTTPS request to a website hosted behind an Application Load Balancer. When the conditions for a rule with an authenticate action are met, the load balancer checks for an authentication session cookie in the request headers.
2. If the cookie is not present, the load balancer redirects the user to the IdP authorization endpoint so that the IdP can authenticate the user.
3. After the user is authenticated, the IdP sends the user back to the load balancer with an authorization grant code.
4. The load balancer presents the authorization grant code to the IdP token endpoint.
5. Upon receiving a valid authorization grant code, the IdP provides the ID token and access token to the Application Load Balancer.
6. The Application Load Balancer then sends the access token to the user info endpoint.
7. The user info endpoint exchanges the access token for user claims.
8. The Application Load Balancer redirects the user with the `AWSELB` authentication session cookie to the original URI. Because most browsers limit the cookie size to 4K, the load balancer shards a cookie that is greater than 4K in size into multiple cookies. If the total size of the user claims and access token received from the IdP is greater than 11K bytes in size, the load balancer returns an HTTP 500 error to the client and increments the `ELBAuthUserClaimsSizeExceeded` metric.
9. The Application Load Balancer validates the cookie and forwards the user info to targets in the `X-AMZN-OIDC-*` HTTP headers set. For more information, see User claims encoding and signature verification.
10. The target sends a response back to the Application Load Balancer.
11. The Application Load Balancer sends the final response to the user.

Every new request goes through steps 1 through 11, while subsequent requests go through steps 9 through 11. That is, every subsequent request starts at step 9 as long as the cookie has not expired.

The `AWSALBAuthNonce` cookie is added to the request header after the user authenticates at the IdP. This does not change how the Application Load Balancer processes redirect requests from the IdP.

If the IdP provides a valid refresh token in the ID token, the load balancer saves the refresh token and uses it to refresh the user claims each time the access token expires, until the session times out or the IdP refresh fails. If the user logs out, the refresh fails and the load balancer redirects the user to the IdP authorization endpoint. This enables the load balancer to drop sessions after the user logs out. For more information, see Session timeout.

Note

The cookie expiry is different from the authentication session expiry. The cookie expiry is an attribute of the cookie, which is set to 7 days. The actual length of the authentication session is determined by the session timeout configured on the Application Load Balancer for the authentication feature. This session timeout is included in the Auth cookie value, which is also encrypted.

User claims encoding and signature verification

After your load balancer authenticates a user successfully, it sends the user claims received from the IdP to the target. The load balancer signs the user claim so that applications can verify the signature and verify that the claims were sent by the load balancer.

The load balancer adds the following HTTP headers:

`x-amzn-oidc-accesstoken`

The access token from the token endpoint, in plain text.

`x-amzn-oidc-identity`

The subject field (`sub`) from the user info endpoint, in plain text.

Note: The sub claim is the best way to identify a given user.

`x-amzn-oidc-data`

The user claims, in JSON web tokens (JWT) format.

Access tokens and user claims are different from ID tokens. Access tokens and user claims only allow access to server resources, while ID tokens carry additional information to authenticate a user. The Application Load Balancer creates a new access token when authenticating a user and

only passes the access tokens and claims to the backend, however it does not pass the ID token information.

These tokens follow the JWT format but are not ID tokens. The JWT format includes a header, payload, and signature that are base64 URL encoded, and includes padding characters at the end. An Application Load Balancer uses ES256 (ECDSA using P-256 and SHA256) to generate the JWT signature.

The JWT header is a JSON object with the following fields:

```
{
   "alg": "algorithm",
   "kid": "12345678-1234-1234-1234-123456789012",
   "signer": "arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/
app/load-balancer-name/load-balancer-id",
   "iss": "url",
   "client": "client-id",
   "exp": "expiration"
}
```

The JWT payload is a JSON object that contains the user claims received from the IdP user info endpoint.

```
{
   "sub": "1234567890",
   "name": "name",
   "email": "alias@example.com",
   ...
}
```

If you want the load balancer to encrypt your user claims you must configure your target group to use HTTPS. Also, as a security best practice we recommend you restrict your targets to only receive traffic from your Application Load Balancer. You can achieve this by configuring your targets' security group to reference the load balancer's security group ID.

To ensure security, you must verify the signature before doing any authorization based on the claims and validate that the `signer` field in the JWT header contains the expected Application Load Balancer ARN.

To get the public key, get the key ID from the JWT header and use it to look up the public key from the endpoint.

AWS provides a library that you can use to verify JWTs signed by Amazon Cognito, Application Load Balancers, and other OIDC-compatible IDPs. For more information, see AWS JWT Verify.

Timeout

Session timeout

The refresh token and the session timeout work together as follows:

- If the session timeout is shorter than the access token expiration, the load balancer honors the session timeout. If the user has an active session with the IdP, the user might not be prompted to log in again. Otherwise, the user is redirected to log in.
  - If the IdP session timeout is longer than the Application Load Balancer session timeout, the user does not have to supply credentials to log in again. Instead, the IdP redirects back to the Application Load Balancer with a new authorization grant code. Authorization codes are single use, even if there is no re-login.
  - If the IdP session timeout is equal to or shorter than the Application Load Balancer session timeout, the user is asked to supply credentials to log in again. After the user logs in, IdP redirects back to the Application Load Balancer with a new authorization grant code, and the rest of the authentication flow continues until the request reaches the backend.
- If the session timeout is longer than the access token expiration and the IdP does not support refresh tokens, the load balancer keeps the authentication session until it times out. Then, it has the user log in again.
- If the session timeout is longer than the access token expiration and the IdP supports refresh tokens, the load balancer refreshes the user session each time the access token expires. The load balancer has the user log in again only after the authentication session times out or the refresh flow fails.

Client login timeout

A client must initiate and complete the authentication process within 15 minutes. If a client fails to complete authentication within the 15-minute limit, it receives an HTTP 401 error from the load balancer. This timeout can't be changed or removed.

For example, if a user loads the login page through the Application Load Balancer, they must complete the login process within 15 minutes. If the user waits and then attempts to log in after

the 15-minute timeout has expired, the load balancer returns an HTTP 401 error. The user will have to refresh the page and attempt logging in again.

Authentication logout

When an application needs to log out an authenticated user, it should set the expiration time of the authentication session cookie to -1 and redirect the client to the IdP logout endpoint (if the IdP supports one). To prevent users from reusing a deleted cookie, we recommend that you configure as short an expiration time for the access token as is reasonable. If a client provides the load balancer with a session cookie that has an expired access token with a non-NULL refresh token, the load balancer contacts the IdP to determine whether the user is still logged in.

Client logout landing pages are unauthenticated. This means that they cannot be behind an Application Load Balancer rule that requires authentication.

- When a request is sent to the target, the application must set the expiry to -1 for all authentication cookies. Application Load Balancers support cookies up to 16K in size and can therefore create up to 4 shards to send to the client.
  - If the IdP has a logout endpoint, it should issue a redirect to the IdP logout endpoint, for example, the LOGOUT Endpoint documented in the *Amazon Cognito Developer Guide*.
  - If the IdP does not have a logout endpoint, the request goes back to the client logout landing page, and the login process is restarted.
- Assuming that the IdP has a logout endpoint, the IdP must expire access tokens and refresh tokens, and redirect the user back to the client logout landing page.
- Subsequent requests follow the original authentication flow.

Verify JWTs using an Application Load Balancer

You can configure an Application Load Balancer (ALB) to verify JSON Web Tokens (JWT) provided by clients for secure service-to-service (S2S) or machine-to-machine (M2M) communications. The load balancer can verify a JWT no matter how it was issued and without human interaction.

ALB will validate the token signature and requires two mandatory claims: 'iss' (issuer) and 'exp' (expiration). Additionally, if present in the token, ALB will also validate 'nbf' (not before) and 'iat' (issued at time) claims. You can configure up to 10 additional claims for validation. These claims support three formats:

- Single-string: A single text value

- Space-separated values: Multiple values separated by spaces (maximum 10 values)
- String-array: An array of text values (maximum 10 values)

If the token is valid, the load balancer forwards the request with token as is to the target. Otherwise, it rejects the request.

Prepare to use JWT verification

Complete the following tasks:

1. Register your service with an IdP, which issues a client ID and a client secret.
2. Make a separate call to the IdP to request access to a service. The IdP responds with an access token. This token is typically a JWT signed by the IdP.
3. Set up a JSON Web Key Sets (JWKS) endpoint. The load balancer acquires the public key published by the IdP in a well-known location that you configure.
4. Include the JWT in a request header, and forward it to the Application Load Balancer in every request.

To configure JWT verification using console

1. Open the Amazon EC2 console console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Load Balancers**.
3. Select your Application Load Balancer and choose the **Listeners** tab.
4. Select an HTTPS listener and choose **Manage rules**.
5. Choose **Add rule**.
6. (Optional) To specify a name for your rule, expand **Name and tags**, and enter the name. To add additional tags, choose **Add additional tags** and enter the tag key and tag value.
7. Under **Conditions**, define 1-5 condition values
8. (Optional) To add a transform, choose **Add transform**, choose the transform type, and enter a regular expression to match and a replacement string.
9. For **Actions, Pre-routing** action, choose **Validate token.**
    a. For **JWKS endpoint**, enter the URL of your JSON Web Key Set endpoint. This endpoint must be publicly accessible and return the public keys used to verify JWT signatures.
    b. For **Issuer**, enter the expected value of the iss claim in your JWT tokens.

c. (Optional) To validate additional claims, choose **Additional claim.**

i. For **Claim name**, enter the name of the claim to validate.

ii. For **Format**, choose how the claim values should be interpreted:

1. **Single string**: The claim must match exactly one specified value.
2. **String array**: The claim must match one of the values in an array.
3. **Space separated values**: The claim contains space-separated values that must include the specified values.

iii. For **Values**, enter the expected values for the claim.

iv. Repeat for additional claims (maximum 10 claims).

10. For **Actions, Routing action**, select the primary action **(Forward to, Redirect to, or Return fixed response)** that should be performed after successful token validation.
11. Configure the primary action as needed
12. Choose **Save.**

To configure JWT verification using CLI

Use the following create-rule command to configure JWT verification .

Create a listener rule with an action to verify JWTs. The listener must be an HTTPS listener.

```
aws elbv2 create-rule \
    --listener-arn listener-arn \
    --priority 10 \
    --conditions Field=path-pattern,Values="/login" \
    --actions file://actions.json
```

The following is an example of the `actions.json` file that specifies a `jwt-validation` action and a `forward` action. Please follow documentation provided by your identity provider to determine the fields that are supported

```
--actions '[
    {
        "Type":"jwt-validation",
        "JwtValidationConfig":{
            "JwksEndpoint":"https://issuer.example.com/.well-known/jwks.json",
```

```
            "Issuer":"https://issuer.com"
        },
        "Order":1
    },
    {
        "Type":"forward",
        "TargetGroupArn":"target-group-arn",
        "Order":2
    }
]'
```

The following example specifies an additional claim to validate.

```
--actions '[
    {
        "Type":"jwt-validation",
        "JwtValidationConfig":{
            "JwksEndpoint":"https://issuer.example.com/.well-known/jwks.json",
            "Issuer":"https://issuer.com",
            "AdditionalClaims":[
              {
                  "Format":"string-array",
                  "Name":"claim_name",
                  "Values":["value1","value2"]
              }
            ],
        },
        "Order":1
    },
    {
        "Type":"forward",
        "TargetGroupArn":"target-group-arn",
        "Order":2
    }
]'
```

For more information, see the section called “Listener rules”.

HTTP headers and Application Load Balancers

HTTP requests and HTTP responses use header fields to send information about the HTTP messages. HTTP headers are added automatically. Header fields are colon-separated name-value

pairs that are separated by a carriage return (CR) and a line feed (LF). A standard set of HTTP header fields is defined in RFC 2616, Message Headers. There are also non-standard HTTP headers available that are automatically added and widely used by the applications. Some of the non-standard HTTP headers have an `X-Forwarded` prefix. Application Load Balancers support the following `X-Forwarded` headers.

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

X-Forwarded headers

- X-Forwarded-For
- X-Forwarded-Proto
- X-Forwarded-Port

X-Forwarded-For

The `X-Forwarded-For` request header helps you identify the IP address of a client when you use an HTTP or HTTPS load balancer. Because load balancers intercept traffic between clients and servers, your server access logs only contain the IP address of the load balancer. To see the IP address of the client, use the `routing.http.xff_header_processing.mode` attribute. This attribute enables you to modify, preserve, or remove the `X-Forwarded-For` header in the HTTP request before the Application Load Balancer sends the request to the target. The possible values for this attribute are append, `preserve`, and `remove`. The default value for this attribute is append.

> **Important**
>
> The `X-Forwarded-For` header should be used with caution due to the potential for security risks. The entries can only be considered trustworthy if added by systems that are properly secured within the network.

Processing mode

- Append
- Preserve
- Remove

Append

By default, the Application Load Balancer stores the IP address of the client in the `X-Forwarded-For` request header and passes the header to your server. If the `X-Forwarded-For` request header is not included in the original request, the load balancer creates one with the client IP address as the request value. Otherwise, the load balancer appends the client IP address to the existing header and then passes the header to your server. The `X-Forwarded-For` request header may contain multiple IP addresses that are comma separated.

The `X-Forwarded-For` request header takes the following form:

```
X-Forwarded-For: client-ip-address
```

The following is an example `X-Forwarded-For` request header for a client with an IP address of `203.0.113.7`.

```
X-Forwarded-For: 203.0.113.7
```

The following is an example `X-Forwarded-For` request header for a client with an IPv6 address of `2001:DB8::21f:5bff:febf:ce22:8a2e`.

```
X-Forwarded-For: 2001:DB8::21f:5bff:febf:ce22:8a2e
```

When the client port preservation attribute (`routing.http.xff_client_port.enabled`) is enabled on the load balancer, the `X-Forwarded-For` request header includes the `client-port-number` appended to the `client-ip-address`, separated by a colon. The header then takes the following form:

```
IPv4 -- X-Forwarded-For: client-ip-address:client-port-number
```

```
IPv6 -- X-Forwarded-For: [client-ip-address]:client-port-number
```

For IPv6, note that when the load balancer appends the `client-ip-address` to the existing header, it encloses the address in square brackets.

The following is an example `X-Forwarded-For` request header for a client with an IPv4 address of `12.34.56.78` and a port number of `8080`.

```
X-Forwarded-For: 12.34.56.78:8080
```

The following is an example `X-Forwarded-For` request header for a client with an IPv6 address of `2001:db8:85a3:8d3:1319:8a2e:370:7348` and a port number of `8080`.

```
X-Forwarded-For: [2001:db8:85a3:8d3:1319:8a2e:370:7348]:8080
```

Preserve

The `preserve` mode in the attribute ensures that the `X-Forwarded-For` header in the HTTP request is not modified in any way before it is sent to targets.

Remove

The `remove` mode in the attribute removes the `X-Forwarded-For` header in the HTTP request before it is sent to targets.

If you enable the client port preservation attribute (`routing.http.xff_client_port.enabled`), and also select `preserve` or `remove` for the `routing.http.xff_header_processing.mode` attribute, the Application Load Balancer overrides the client port preservation attribute. It keeps the `X-Forwarded-For` header unchanged, or removes it depending on the mode you select, before it sends it to the targets.

The following table shows examples of the `X-Forwarded-For` header that the target receives when you select either the append, `preserve` or the `remove` mode. In this example, the IP address of the last hop is `127.0.0.1`.

Request description	Example request	append	preserve	remove
Request is sent with no XFF header	`GET / index.html HTTP/1.1 Host: example.com`	`X-Forwarded-For: 127.0.0.1`	Not present	Not present
Request is sent with an XFF	`GET / index.html HTTP/1.1`	`X-Forwarded-For:`	`X-Forwarded-For: 127.0.0.4`	Not present

Request description	Example request	append	preserve	remove
header and a client IP address.	`Host: example.com X-Forwarded-For: 127.0.0.4`	`127.0.0.4, 127.0.0.1`		
Request is sent with an XFF header with multiple client IP addresses.	`GET /index.html HTTP/1.1 Host: example.com X-Forwarded-For: 127.0.0.4, 127.0.0.8`	`X-Forwarded-For: 127.0.0.4, 127.0.0.8, 127.0.0.1`	`X-Forwarded-For: 127.0.0.4, 127.0.0.8`	Not present

Console

To manage the X-Forwarded-For header

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Attributes** tab, choose **Edit**.
5. In the **Traffic configuration** section, under **Packet handling**, for **X-Forwarded-For header**, choose **Append** (default), **Preserve**, or **Remove**.
6. Choose **Save changes**.

AWS CLI

To manage the X-Forwarded-For header

Use the modify-load-balancer-attributes command with the `routing.http.xff_header_processing.mode` attribute. The possible values are `append`, `preserve`, and `remove`. The default is `append`.

```
aws elbv2 modify-load-balancer-attributes \
    --load-balancer-arn load-balancer-arn \
    --attributes "Key=routing.http.xff_header_processing.mode,Value=preserve"
```

CloudFormation

To manage the X-Forwarded-For header

Update the AWS::ElasticLoadBalancingV2::LoadBalancer resource to include the `routing.http.xff_header_processing.mode` attribute. The possible values are `append`, `preserve`, and `remove`. The default is `append`.

```
Resources:
  myLoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "routing.http.xff_header_processing.mode"
          Value: "preserve"
```

X-Forwarded-Proto

The `X-Forwarded-Proto` request header helps you identify the protocol (HTTP or HTTPS) that a client used to connect to your load balancer. Your server access logs contain only the protocol used between the server and the load balancer; they contain no information about the protocol used between the client and the load balancer. To determine the protocol used between the client and the load balancer, use the `X-Forwarded-Proto` request header. ELB stores the protocol used

between the client and the load balancer in the `X-Forwarded-Proto` request header and passes the header along to your server.

Your application or website can use the protocol stored in the `X-Forwarded-Proto` request header to render a response that redirects to the appropriate URL.

The `X-Forwarded-Proto` request header takes the following form:

```
X-Forwarded-Proto: originatingProtocol
```

The following example contains an `X-Forwarded-Proto` request header for a request that originated from the client as an HTTPS request:

```
X-Forwarded-Proto: https
```

X-Forwarded-Port

The `X-Forwarded-Port` request header helps you identify the destination port that the client used to connect to the load balancer.

HTTP header modification for your Application Load Balancer

HTTP header modification is supported by Application Load Balancers, for both request and response headers. Without having to update your application code, header modification allows you more control over your application's traffic and security.

To enable header modification, see Enable header modification.

Rename mTLS/TLS headers

The header rename capability allows you to configure the names of the mTLS and TLS headers that the Application Load Balancer generates and adds to requests.

This ability to modify HTTP headers enables your Application Load Balancer to easily support applications that use specifically formatted request and response headers.

Header	Description
X-Amzn-Mtls-Clientcert-Serial-Number	Ensures that the target can identify and verify the specific certificate presented by the client during the TLS handshake.
X-Amzn-Mtls-Clientcert-Issuer	Helps the target validate and authenticate the client certificate by identifying the certificate authority that issued the certificate.
X-Amzn-Mtls-Clientcert-Subject	Provides the target with detailed informati on about the entity the client certificate was issued to, which helps in identification, authentication, authorization, and logging during mTLS authentication.
X-Amzn-Mtls-Clientcert-Validity	Allows the target to verify that the client certificate being used is within its defined validity period, ensuring the certificate is not expired or prematurely used.
X-Amzn-Mtls-Clientcert-Leaf	Provides the client certificate used in the mTLS handshake, allowing the server to authenticate the client and validate the certificate chain. This ensures the connection is secure and authorized.
X-Amzn-Mtls-Clientcert	Carries the full client certificate. Allowing the target to verify the certificate's authenticity, validate the certificate chain, and authenticate the client during the mTLS handshake process.
X-Amzn-TLS-Version	Indicates the version of the TLS protocol used for a connection. It facilitates determining the security level of the communication, troublesh oot connection issues and ensuring complianc e.

Header	Description
X-Amzn-TLS-Cipher-Suite	Indicates the combination of cryptographic algorithms used to secure a connection in TLS. This allows the server to assess the security of the connection, helping with compatibility troubleshooting, and ensuring compliance with security policies.

Add response headers

Using insert headers, you can configure your Application Load Balancer to add security-related headers to responses. With these attributes, you can insert headers including HSTS, CORS, and CSP.

By default, these headers are empty. When this happens, the Application Load Balancer does not modify this response header.

When you enable a response header, the Application Load Balancer adds the header with the configured value to all responses. If the response from target includes the HTTP response header, the load balancer updates the header value to be the configured value. Otherwise, the load balancer adds the HTTP response header to the response with the configured value.

Header	Description
Strict-Transport-Security	Enforces HTTPS-only connections by the browser for a specified duration, helping to protect against man-in-the-middle attacks, protocol downgrades and user errors. ensuring all communications between the client and target is encrypted.
Access-Control-Allow-Origin	Controls whether resources on a target can be accessed from different origins. This allows secure cross-origin interactions while preventing unauthorized access.
Access-Control-Allow-Methods	Specifies the HTTP methods that are allowed when making cross-origin requests to the

Header	Description
	target. It provides control over which actions can be performed from different origins.
Access-Control-Allow-Headers	Specifies which custom or non-simple headers can be included in a cross-origin request. This header gives targets control over which headers can be sent by clients from different origins.
Access-Control-Allow-Credentials	Specifies whether the client should include credentials such as cookies, HTTP authentic ation or client certificates in cross-origin requests.
Access-Control-Expose-Headers	Allows the target to specify which additional response headers can be access by the client in cross-origin requests.
Access-Control-Max-Age	Defines how long the browser can cache the result of a preflight request, reducing the need for repeated preflight checks. This helps to optimize performance by reducing the number of OPTIONS requests required for certain cross-origin requests.
Content-Security-Policy	Security feature that prevents code injection attacks like XSS by controlling which resources such as scripts, styles, images, etc. can be loaded and executed by a website.
X-Content-Type-Options	With the no-sniff directive, enhances web security by preventing browsers from guessing the MIME type of a resource. It ensures that browsers only interpret content according to the declared Content-Type

Header	Description
X-Frame-Options	Header security mechanism that helps prevent click-jacking attacks by controlling whether a web page can be embedded in frames. Values such as DENY and SAMEORIGIN can ensure that content is not embedded on malicious or untrusted websites.

Disable headers

Using disable headers, you can configure your Application Load Balancer to disable the `server:awselb/2.0` header from the responses. This reduces exposure of server specific information, while adding an extra layer of protection to your application.

The attribute name is `routing.http.response.server.enabled`. The available values are `true` or `false`. The default value is `true`.

Limitations

- Header values can contain the following characters
  - Alphanumeric characters: `a-z`, `A-Z`, and `0-9`
  - Special characters: `_ :;.,\/'?!(){}[]@<>=-+*#&`|~^%`
- The value for the attribute can not exceed 1K bytes in size.
- ELB performs basic input validations to verify the header value is valid. However the validation is unable to confirm if the value is supported for a specific header.
- Setting an empty value for any attribute will cause the Application Load Balancer to revert to the default behavior.

Enable HTTP header modification for your Application Load Balancer

Header modification is turned off by default and must be enabled on each listener. For more information, see HTTP header modification.

Console

To enable header modification

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the Application Load Balancer.
4. On the **Listeners and rules** tab, select the protocol and port to open the details page for your listener.
5. On the **Attributes** tab, select **Edit**.

   Listener attributes are organized into groups. You'll choose which features to enable.
6. [HTTPS listeners] **Modifiable mTLS/TLS header names**

   a. Expand **Modifiable mTLS/TLS header names**.

   b. Enable the request headers to modify and provide names for them. For more information, see the section called "Rename mTLS/TLS headers".
7. **Add response headers**

   a. Expand **Add response headers**.

   b. Enable the response headers to add and provide values for them. For more information, see the section called "Add response headers".
8. **ALB server response header**

   - Enable or disable **Server header**.
9. Choose **Save changes**.

AWS CLI

To enable header modification

Use the modify-listener-attributes command. For the list of attributes, see the section called "Header modification attributes".

```
aws elbv2 modify-listener-attributes \
    --listener-arn listener-arn \
    --attributes "Key=attribute-name,Value=attribute-value"
```

CloudFormation

To enable header modification

Update the AWS::ElasticLoadBalancingV2::Listener resource to include the attributes. For the list of attributes, see the section called "Header modification attributes".

```
Resources:
  myHTTPlistener:
  Type: 'AWS::ElasticLoadBalancingV2::Listener'
  Properties:
    LoadBalancerArn: !Ref myLoadBalancer
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - Type: "forward"
        TargetGroupArn: !Ref myTargetGroup
    ListenerAttributes:
      - Key: "attribute-name"
        Value: "attribute-value"
```

Header modification attributes

The following are the header modification attributes supported by Application Load Balancers.

`routing.http.request.x_amzn_mtls_clientcert_serial_number.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert-Serial-Number**.

`routing.http.request.x_amzn_mtls_clientcert_issuer.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert-Issuer**.

`routing.http.request.x_amzn_mtls_clientcert_subject.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert-Subject**.

`routing.http.request.x_amzn_mtls_clientcert_validity.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert-Validity**.

`routing.http.request.x_amzn_mtls_clientcert_leaf.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert-Leaf**.

`routing.http.request.x_amzn_mtls_clientcert.header_name`

Modify the header name of **X-Amzn-Mtls-Clientcert**.

`routing.http.request.x_amzn_tls_version.header_name`

Modify the header name of **X-Amzn-Tls-Version**.

`routing.http.request.x_amzn_tls_cipher_suite.header_name`

Modify the header name of **X-Amzn-Tls-Cipher-Suite**.

`routing.http.response.server.enabled`

Indicates whether to allow or remove the HTTP response server header.

`routing.http.response.strict_transport_security.header_value`

Add the **Strict-Transport-Security** header to inform browsers that the site should only be accessed using HTTPS, and that any future attempts to access it using HTTP should automatically be converted to HTTPS.

`routing.http.response.access_control_allow_origin.header_value`

Add the **Access-Control-Allow-Origin** header to specify which origins are allowed to access the server.

`routing.http.response.access_control_allow_methods.header_value`

Add the **Access-Control-Allow-Methods** header to specify which HTTP methods are allowed when accessing the server from a different origin.

`routing.http.response.access_control_allow_headers.header_value`

Add the **Access-Control-Allow-Headers** header to specify which headers are allowed during a cross-origin request.

`routing.http.response.access_control_allow_credentials.header_value`

Add the **Access-Control-Allow-Credentials** header to indicate whether the browser should include credentials such as cookies or authentication in cross-origin requests.

`routing.http.response.access_control_expose_headers.header_value`

Add the **Access-Control-Expose-Headers** header to indicate which headers the browser can expose to the requesting client.

`routing.http.response.access_control_max_age.header_value`

Add the **Access-Control-Max-Age** header to specify how long the results of a preflight request can be cached, in seconds.

`routing.http.response.content_security_policy.header_value`

Add the **Content-Security-Policy** header to specify restrictions enforced by the browser to help minimize the risk of certain types of security threats.

`routing.http.response.x_content_type_options.header_value`

Add the **X-Content-Type-Options** header to indicate whether the MIME types advertised in the **Content-Type** headers should be followed and not be changed.

`routing.http.response.x_frame_options.header_value`

Add the **X-Frame-Options** header to indicate whether the browser is allowed to render a page in a **frame**, **iframe**, **embed**, or **object**.

Delete a listener for your Application Load Balancer

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

- The load balancer immediately stops accepting new connections on the listener port.
- Active connections are closed. Any requests in progress when the listener is deleted will likely fail.

Console

To delete a listener

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, choose **Load Balancers**.
3. Select the load balancer.
4. On the **Listeners and rules** tab, select the check box for the listener and choose **Manage listener**, **Delete listener**.
5. When prompted for confirmation, enter **`confirm`** and then 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 Application Load Balancers

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

Each target group is used to route requests to one or more registered targets. When you create each listener rule, you specify a target group and conditions. When a rule condition is met, traffic is forwarded to the corresponding target group. 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. You can use each target group with only one load balancer. For more information, see Application 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.

Contents

- Use Lambda functions as targets of an Application Load Balancer
- Tags for your Application Load Balancer target group
- Delete an Application Load Balancer target group

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 support the following protocols and ports:

- **Protocols**: HTTP, HTTPS
- **Ports**: 1-65535

When a target group is configured with the HTTPS protocol or uses HTTPS health checks, if any HTTPS listener is using a TLS 1.3 security policy, the `ELBSecurityPolicy-TLS13-1-0-2021-06` security policy will be used for target connections. Otherwise, the `ELBSecurityPolicy-2016-08` security policy is used. 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, and its targets are 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. Traffic that leaves AWS will not have these same protections, and additional steps may be needed to secure traffic further.

Target type

When you create a target group, you specify its target type, which determines the type of target you specify when registering targets with this target group. 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 IP addresses.

`lambda`

The target is a Lambda function.

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

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

Important

You can't specify publicly routable IP addresses.

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

- Instances in a VPC that is peered to the load balancer VPC (same Region or different Region).
- AWS resources that are addressable by IP address and port (for example, databases).
- On-premises resources linked to AWS through Direct Connect or a Site-to-Site VPN connection.

Note

For Application Load Balancers deployed within a Local Zone, the `ip` targets must be in the same Local Zone to receive traffic.
For more information, see What is AWS Local Zones?

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance. 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. Each network interface can have its own security group.

If the target type of your target group is `lambda`, you can register a single Lambda function. When the load balancer receives a request for the Lambda function, it invokes the Lambda function. For more information, see Use Lambda functions as targets of an Application Load Balancer.

You can configure Amazon Elastic Container Service (Amazon ECS) as a target of your Application Load Balancer. For more information, see Use an Application Load Balancer for Amazon ECS in the *Amazon Elastic Container Service Developer Guide*.

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 Application 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 register a Lambda function with an IPv6 target group.

Protocol version

By default, Application Load Balancers send requests to targets using HTTP/1.1. You can use the protocol version to send requests to targets using HTTP/2 or gRPC.

The following table summarizes the result for the combinations of request protocol and target group protocol version.

Request protocol	Protocol version	Result
HTTP/1.1	HTTP/1.1	Success
HTTP/2	HTTP/1.1	Success
gRPC	HTTP/1.1	Error
HTTP/1.1	HTTP/2	Error
HTTP/2	HTTP/2	Success
gRPC	HTTP/2	Success if targets support gRPC
HTTP/1.1	gRPC	Error
HTTP/2	gRPC	Success if a POST request
gRPC	gRPC	Success

Considerations for the gRPC protocol version

- The only supported listener protocol is HTTPS.
- The only supported action type for listener rules is `forward`.
- The only supported target types are `instance` and `ip`.
- The load balancer parses gRPC requests and routes the gRPC calls to the appropriate target groups based on the package, service, and method.
- The load balancer supports unary, client-side streaming, server-side streaming, and bi-directional streaming.
- You must provide a custom health check method with the format `/package.service/method`.
- You must specify the gRPC status codes to use when checking for a successful response from a target.
- You can't use Lambda functions as targets.

Considerations for the HTTP/2 protocol version

- The only supported listener protocol is HTTPS.
- The only supported action type for listener rules is `forward`.
- The only supported target types are `instance` and `ip`.
- The load balancer supports unary, client-side streaming, server-side streaming, and bi-directional streaming. The maximum number of streams per client HTTP/2 connection is 128.

Registered targets

Your load balancer serves as a single point of contact for clients and distributes incoming traffic across its healthy registered targets. 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 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 requests 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 requests.

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*.

Limits

- You can't register the IP addresses of another Application Load Balancer in the same VPC. If the other Application Load Balancer is in a VPC that is peered to the load balancer VPC, you can register its IP addresses.
- You can't register instances by instance ID if they are in a VPC that is peered to the load balancer VPC (same Region or different Region). You can register these instances by IP address.

Target Optimizer

You can enable target optimizer on a target group. Target optimizer lets you accurately enforce a maximum number of concurrent requests on a target. It works with the help of an agent that you install and configure on targets. To enable target optimizer, you specify a target control port for the target group. This port is used for management traffic between the agents and load balancer. Target optimizer can only be enabled during target group creation. Target control port once specified cannot be modified. For more information, see the section called “Target Optimizer”.

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 if the target group type is `instance` or `ip`:

`deregistration_delay.timeout_seconds`

The amount of time for ELB to wait before deregistering a target. The range is 0–3600 seconds. The default value is 300 seconds.

`load_balancing.algorithm.type`

The routing algorithm determines how the load balancer selects targets when routing requests. The value is `round_robin`, `least_outstanding_requests`, or `weighted_random`. The default is `round_robin`.

`load_balancing.algorithm.anomaly_mitigation`

Only available when `load_balancing.algorithm.type` is `weighted_random`. Indicates whether anomaly mitigation is enabled. The value is on or `off`. The default is `off`.

`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`.

`slow_start.duration_seconds`

The time period, in seconds, during which the load balancer sends a newly registered target a linearly increasing share of the traffic to the target group. The range is 30–900 seconds (15 minutes). The default is 0 seconds (disabled).

`stickiness.enabled`

Indicates whether sticky sessions are enabled. The value is `true` or `false`. The default is `false`.

`stickiness.app_cookie.cookie_name`

The name of the application cookie. The application cookie name can't have the following prefixes: `AWSALB`, `AWSALBAPP`, or `AWSALBTG`; they're reserved for use by the load balancer.

`stickiness.app_cookie.duration_seconds`

The application-based cookie expiration period, in seconds. After this period, the cookie is considered stale. The minimum value is 1 second and the maximum value is 7 days (604800 seconds). The default value is 1 day (86400 seconds).

`stickiness.lb_cookie.duration_seconds`

The duration-based cookie expiration period, in seconds. After this period, the cookie is considered stale. The minimum value is 1 second and the maximum value is 7 days (604800 seconds). The default value is 1 day (86400 seconds).

`stickiness.type`

The type of stickiness. The possible values are `lb_cookie` and `app_cookie`.

`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 node as unhealthy in DNS, so that traffic is routed only to healthy nodes. 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 node 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 node as unhealthy in DNS, so that traffic is routed only to healthy nodes. 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 node 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 range is 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`.

The following target group attribute is supported if the target group type is `lambda`:

`lambda.multi_value_headers.enabled`

Indicates whether the request and response headers exchanged between the load balancer and the Lambda function include arrays of values or strings. The possible values are `true` or `false`. The default value is `false`. For more information, see Multi-value headers.

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 enable target optimizer on the target group, we recommend you set the health check port of the target group to be the same as the port in TARGET_CONTROL_DATA_ADDRESS. This ensures that the target will fail health checks if the agent is unhealthy. For more information, see the section called “Target Optimizer”.
- You can't use this feature with target groups where the target is a Lambda function. If the Application Load Balancer is the target of a Network Load Balancer or Global Accelerator, do not configure a threshold for DNS failover.
- 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 an Application Load Balancer and one target group is unhealthy in a zone, DNS health checks succeed if at least one other 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.

Monitoring

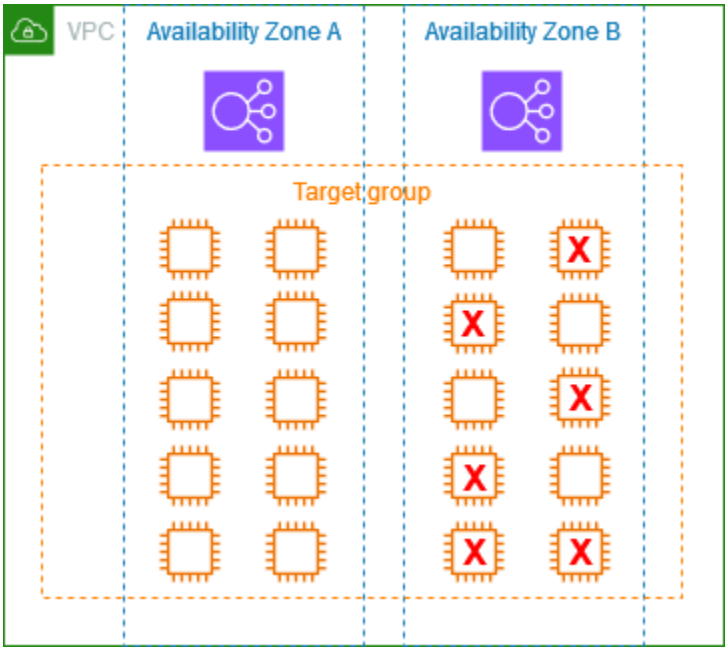
To monitor the health of your target groups, see CloudWatch metrics for target group health.

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 an Application 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 an Application 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 any of the target groups attached to an Application Load Balancer are unhealthy, the alias record fails the Route 53 health check (fail-open). If using evaluate target health, the failover routing policy redirects traffic to the secondary resource.
- If all target groups attached to an Application Load Balancer are empty (no targets), Route 53 considers the record unhealthy (fail-open). If using evaluate target health, the failover routing policy redirects 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 Application Load Balancer

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

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

To route traffic to the targets in a target group, specify the target group in an action when you create a listener or create a rule for your listener. For more information, see Listener rules for your Application Load Balancer. You can specify the same target group in multiple listeners, but these listeners must belong to the same Application 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 with your Application Load Balancer target group. You can also modify the health check settings for your target group. For more information, see Update the health check settings of an Application Load Balancer target group.

Console

To create a target group

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Choose a target type**, select **Instances** to register targets by instance ID, **IP addresses** to register targets by IP address, or **Lambda function** to register a Lambda function as a target.
5. For **Target group name**, type a name for the target group. This name must be unique per region per account, can have a maximum of 32 characters, must contain only alphanumeric characters or hyphens, and must not begin or end with a hyphen.
6. (Optional) For **Protocol** and **Port**, modify the default values as needed.

7. If the target type is **Instances** or **IP addresses**, choose **IPv4** or **IPv6** as the **IP address type**, otherwise skip to the next step.

   Note that only targets that have the selected IP address type can be included in this target group. The IP address type can't be changed after the target group is created.

8. For **VPC**, select a virtual private cloud (VPC). Note that for **IP addresses** target types, the VPCs available for selection are those that support the **IP address type** that you chose in the previous step.

9. (Optional) For **Protocol version**, modify the default value as needed. For more information, see the section called “Protocol version”.

10. (Optional) In the **Health checks** section, modify the default settings as needed. For more information, see the section called “Health check settings”.

11. If the target type is **Lambda function**, you can enable health checks by selecting **Enable** in the **Health checks** section.

12. (Optional) To enable **Target optimizer** on the target group, specify a target control port. The port cannot be modified after target group creation. Target optimizer works with the help of an agent that you install on targets. For more information, see the section called “Target Optimizer”.

13. (Optional) Add one or more tags as follows:

    a. Expand the **Tags** section.

    b. Choose **Add tag**.

    c. Enter the tag key and the tag value.

14. Choose **Next**.

15. (Optional) Add one or more targets as follows:

    - If the target type is **Instances**, select one or more instances, enter one or more ports, and then choose **Include as pending below**.

      Note: The instances must have an assigned primary IPv6 address to be registered with a IPv6 target group.

    - If the target type is **IP addresses**, do the following:

      a. Select a network **VPC** from the list, or choose **Other private IP addresses**.

      b. Enter the IP address manually, or find the IP address using instance details. You can enter up to five IP addresses at a time.

c. Enter the ports for routing traffic to the specified IP addresses.

d. Choose **Include as pending below**.

- If the target type is a **Lambda function**, specify a single Lambda function or omit this step and specify a Lambda function later.

16. 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 HTTP protocol, targets registered by IP address, one tag, and default health check settings.

```
aws elbv2 create-target-group \
    --name my-target-group \
    --protocol HTTP \
    --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 HTTP 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: HTTP
      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
```

Health checks for Application Load Balancer target groups

Your Application Load Balancer periodically sends requests to its registered targets to test their status. These tests are called *health checks*.

Each load balancer node routes requests only to the healthy targets in the enabled Availability Zones for the load balancer. Each load balancer node checks the health of each target, using the health check settings for the target groups with which the target is registered. After your target is registered, it must pass one health check to be considered healthy. After each health check is completed, the load balancer node closes the connection that was established for the health check.

If a target group contains only unhealthy registered targets, the load balancer routes requests to all those targets, regardless of their health status. This means that if all targets fail health checks at the same time in all enabled Availability Zones, the load balancer fails open. The effect of the fail-open is to allow traffic to all targets in all enabled Availability Zones, regardless of their health status, based on the load balancing algorithm.

Health checks do not support WebSockets.

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

You can use health check logs to capture detailed information about the health checks made to your registered targets for your load balancer and store them as log files in Amazon S3. You can use these health check logs to troubleshoot issues with your targets. For more information, see Health check logs.

Contents

- Health check reason codes
- Check the health of your Application Load Balancer targets
- Update the health check settings of an Application Load Balancer target group

Health check settings

You configure health checks for the targets in a target group as described in the following table. The setting names used in the table are the names used in the API. The load balancer sends a health check request to each registered target every **HealthCheckIntervalSeconds** seconds, using the specified port, protocol, and health check path. Each health check request is independent and the result lasts for the entire interval. The time that it takes for the target to respond does not affect the interval for the next health check request. 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.

Note that when you deregister a target, this decreases **HealthyHostCount** but does not increase **UnhealthyHostCount**.

Setting	Description
HealthCheckProtocol	The protocol the load balancer uses when performing health checks on targets. For Application Load Balancers the possible protocols are HTTP and HTTPS. The default is the HTTP protocol. These protocols use the HTTP GET method to send health check requests.
HealthCheckPort	The port the load balancer uses when performing health checks on targets. The default is to use the port on which each target receives traffic from the load balancer.
HealthCheckPath	The destination for health checks on the targets.

Setting	Description
	If the protocol version is HTTP/1.1 or HTTP/2, specify a valid URI (/*path*?*query*). The default is /. If the protocol version is gRPC, specify the path of a custom health check method with the format `/package.service/method` . The default is `/AWS.ALB/healthcheck` .
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 is 5 seconds if the target type is `instance` or `ip` and 30 seconds if the target type is `lambda`.
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 if the target type is `instance` or `ip` and 35 seconds if the target type is `lambda`.
HealthyThresholdCount	The number of consecutive successful health checks required before considering an unhealthy target healthy. The range is 2–10. The default is 5.
UnhealthyThresholdCount	The number of consecutive failed health checks required before considering a target unhealthy. The range is 2–10. The default is 2.

Setting	Description
Matcher	The codes to use when checking for a successful response from a target. These are called **Success codes** in the console. If the protocol version is HTTP/1.1 or HTTP/2, the possible values are from 200 to 499. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299"). The default value is 200. If the protocol version is gRPC, the possible values are from 0 to 99. You can specify multiple values (for example, "0,1") or a range of values (for example, "0-5"). The default value is 12.

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. Before a target can receive requests from the load balancer, it must pass the initial health checks. After a target passes the initial health checks, its status is `Healthy`.

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. Related reason codes: `Elb.RegistrationInProgress` | `Elb.InitialHealthChecking`
`healthy`	The target is healthy.

Value	Description
	Related reason codes: None
`unhealthy`	The target did not respond to a health check or failed the health check. Related reason codes: `Target.ResponseCodeMismatch` | `Target.Timeout` | `Target.FailedHealthChecks` | `Elb.InternalError`
`unused`	The target is not registered with a target group, the target group is not used in a listener rule, the target is in an Availability Zone that is not enabled, or the target is in the stopped or terminated state. Related reason codes: `Target.NotRegistered` | `Target.NotInUse` | `Target.InvalidState` | `Target.IpUnusable`
`draining`	The target is deregistering and connection draining is in process. Related reason code: `Target.DeregistrationInProgress`
`unavailable`	Health checks are disabled for the target group. Related reason code: `Target.HealthCheckDisabled`

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. Reason codes that begin with `Elb` originate on the load balancer side and reason codes that begin with `Target` originate on the target side. For more information about possible causes for health check failures, see Troubleshooting.

Reason code	Description
`Elb.InitialHealthChecking`	Initial health checks in progress
`Elb.InternalError`	Health checks failed due to an internal error
`Elb.RegistrationInProgress`	Target registration is in progress
`Target.DeregistrationInProgress`	Target deregistration is in progress
`Target.FailedHealthChecks`	Health checks failed
`Target.HealthCheckDisabled`	Health checks are disabled
`Target.InvalidState`	Target is in the stopped state Target is in the terminated state Target is in the terminated or stopped state 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 Target is in an Availability Zone that is not enabled for the load balancer
`Target.NotRegistered`	Target is not registered to the target group
`Target.ResponseCodeMismatch`	Health checks failed with these codes: [*code*]
`Target.Timeout`	Request timed out

Check the health of your Application 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.

You can use health check logs to capture detailed information about the health checks made to your registered targets for your load balancer and store them as log files in Amazon S3. You can use these health check logs to troubleshoot issues with your targets. For more information, see Health check logs.

Console

To check the health of your targets

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. The **Details** tab displays the total number of targets, plus the number of targets for each health status.
5. On the **Targets** tab, the **Status** column indicates the status of each target.
6. If the status is any value other than `Healthy`, the **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.ResponseCodeMismatch` - The health checks did not return an expected HTTP code.
- `Target.Timeout` - The health check requests timed out.
- `Target.FailedHealthChecks` - The load balancer received an error while establishing a connection to the target or the target response was malformed.
- `Elb.InternalError` - The health checks failed due to an internal error.

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

- `Target.HealthCheckDisabled` - Health checks are disabled for the target group.

Update the health check settings of an Application 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://eusc-de-east-1.console.amazonaws-eusc.eu/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 **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: HTTP
      Port: 80
      TargetType: instance
      VpcId: !Ref myVPC
      HealthyThresholdCount: 3
      HealthCheckTimeoutSeconds: 20
```

Edit target group attributes for your Application Load Balancer

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

Target group attributes

- Deregistration delay
- Routing algorithm
- Slow start mode
- Health settings
- Cross-zone load balancing
- Automatic Target Weights (ATW)
- Sticky sessions

Deregistration delay

ELB stops sending requests to targets that are deregistering. By default, ELB waits 300 seconds before completing the deregistration process, which can help in-flight requests to the target to complete. To change the amount of time that ELB waits, update the deregistration delay value.

The initial state of a deregistering target is `draining`. After the deregistration delay elapses, the deregistration process completes and the state of the target is `unused`. If the target is part of an Auto Scaling group, it can be terminated and replaced.

If a deregistering target has no in-flight requests and no active connections, ELB immediately completes the deregistration process, without waiting for the deregistration delay to elapse. However, even though target deregistration is complete, the status of the target is displayed as `draining` until the deregistration delay timeout expires. After the timeout expires, the target transitions to an `unused` state.

If a deregistering target terminates the connection before the deregistration delay elapses, the client receives a 500-level error response.

Console

To update the deregistration delay value

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. In the **Target deregistration management** pane, enter a new value for **Deregistration delay**.
6. Choose **Save changes**.

AWS CLI

To update the deregistration delay value

Use the modify-target-group-attributes command with the `deregistration_delay.timeout_seconds` attribute.

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

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

CloudFormation

To update the deregistration delay value

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `deregistration_delay.timeout_seconds` attribute.

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

Routing algorithm

A routing algorithm is a method used by the load balancer when determining which targets will receive requests. The **round robin** routing algorithm is used by default to route requests at the target group level. The **least outstanding requests** and **weighted random** routing algorithms are also available based on the needs of your application. A target group can only have one active routing algorithm at a time, however the routing algorithm can be updated whenever needed.

If you enable sticky sessions, the selected routing algorithm is used for the initial target selection. Future requests from the same client will be forwarded to the same target, bypassing the selected routing algorithm. If you have enabled target optimizer, the routing algorithm can only be round robin.

Round robin

- The round robin routing algorithm routes requests evenly across healthy targets in the target group, in a sequential order.

- This algorithm is commonly used when the requests being received are similar in complexity , the registered targets are similar in processing capability, or if you need to distribute requests equally among targets.

Least outstanding requests

- The least outstanding requests routing algorithm routes requests to the targets with the lowest number of in progress requests.
- This algorithm is commonly used when the requests being received vary in complexity, the registered targets vary in processing capability.
- When a load balancer that supports HTTP/2 is using targets that only support HTTP/1.1, it converts the request to multiple HTTP/1.1 requests. In this configuration the least outstanding requests algorithm will treat each HTTP/2 request as multiple requests.
- When using WebSockets, the target is selected using the least outstanding requests algorithm. After the target is selected, the load balancer creates a connection to the target and sends all messages over this connection.
- The least outstanding requests routing algorithm can not be used with slow start mode.

Weighted random

- The weighted random routing algorithm routes requests evenly across healthy targets in the target group, in a random order.
- This algorithm supports Automatic Target Weights (ATW) anomaly mitigation.
- The weighted random routing algorithm can not be used with slow start mode.
- The weighted random routing algorithm can not be used with sticky sessions.

Console

To update the routing algorithm

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. In the **Traffic configuration** pane, for **Load balancing algorithm**, choose **Round robin**, **Least outstanding requests**, or **Weighted random**.
6. Choose **Save changes**.

AWS CLI

To update the routing algorithm

Use the modify-target-group-attributes command with the `load_balancing.algorithm.type` attribute.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes
 "Key=load_balancing.algorithm.type,Value=least_outstanding_requests"
```

CloudFormation

To update the routing algorithm

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `load_balancing.algorithm.type` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "load_balancing.algorithm.type"
          Value: "least_outstanding_requests"
```

Slow start mode

By default, a target starts to receive its full share of requests as soon as it is registered with a target group and passes an initial health check. Using slow start mode gives targets time to warm up before the load balancer sends them a full share of requests.

After you enable slow start for a target group, its targets enter slow start mode when they are considered healthy by the target group. A target in slow start mode exits slow start mode when the configured slow start duration period elapses or the target becomes unhealthy. The load balancer linearly increases the number of requests that it can send to a target in slow start mode. After a healthy target exits slow start mode, the load balancer can send it a full share of requests.

Considerations

- When you enable slow start for a target group, the healthy targets registered with the target group do not enter slow start mode.
- When you enable slow start for an empty target group and then register targets using a single registration operation, these targets do not enter slow start mode. Newly registered targets enter slow start mode only when there is at least one healthy target that is not in slow start mode.
- If you deregister a target in slow start mode, the target exits slow start mode. If you register the same target again, it enters slow start mode when it is considered healthy by the target group.
- If a target in slow start mode becomes unhealthy, the target exits slow start mode. When the target becomes healthy, it enters slow start mode again.
- You can't enable slow start mode when using the **least outstanding requests** or **weighted random** routing algorithms.

Console

To update the slow start duration value

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. In the **Traffic configuration** pane, enter a new value for **Slow start duration**. To disable slow start mode, enter 0.

6. Choose **Save changes**.

AWS CLI

To update the slow start duration value

Use the modify-target-group-attributes command with the `slow_start.duration_seconds` attribute.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes "Key=slow_start.duration_seconds,Value=30"
```

CloudFormation

To update the slow start duration value

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `slow_start.duration_seconds` attribute.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "slow_start.duration_seconds"
          Value: "30"
```

Health settings

By default, Application 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 modify target group health settings

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. Check whether cross-zone load balancing is turned on or turned off. Update this setting as needed to ensure that you have enough capacity to handle the additional traffic if a zone fails.
6. Expand **Target group health requirements**.
7. For **Configuration type**, we recommend that you choose **Unified configuration**, which sets the same threshold for both actions.
8. 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.
9. Choose **Save changes**.

AWS CLI

To modify 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: HTTP
      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"
```

Cross-zone load balancing

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 only across the registered targets in its Availability Zone. This could be 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 Application Load Balancers, cross-zone load balancing is always turned on at the load balancer level, and cannot be turned off. 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 off at the target group level.

Considerations

- Target stickiness is not supported when cross-zone load balancing is off.

- Lambda functions as targets are not supported when cross-zone load balancing is off.
- Attempting to turn off cross-zone load balancing through the `ModifyTargetGroupAttributes` API if any targets have parameter `AvailabilityZone` set to `all` results in an error.
- When registering targets, the `AvailabilityZone` parameter is required. Specific Availability Zone values are only allowed when cross-zone load balancing is off. Otherwise, the parameter is ignored and treated as `all`.

Best practices

- Plan for enough target capacity across all Availability Zones that you expect to utilize, per target group. If you can't plan for enough capacity across all participating Availability Zones, we recommend that you keep cross-zone load balancing on.
- When configuring your Application Load Balancer with multiple target groups, ensure all target groups are participating in the same Availability Zones, within the configured Region. This is to avoid an Availability Zone being empty while cross-zone load balancing is off, as this triggers a 503 error for all HTTP requests that enter the empty Availability Zone.
- Avoid creating empty subnets. Application Load Balancers expose zonal IP addresses through DNS for the empty subnets, which triggers 503 errors for HTTP requests.
- There can be occurrences where a target group with cross-zone load balancing turned off has enough planned target capacity per Availability Zone, but all targets in an Availability Zone become unhealthy. When there is at least one target group with all unhealthy targets, the IP addresses of the load balancer nodes are removed from DNS. After the target group has at least one healthy target, the IP addresses are restored to DNS.

Turn off cross-zone load balancing

You can turn off cross-zone load balancing for your Application Load Balancer target groups at any time.

Console

To turn off cross-zone load balancing

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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, select **Edit**.
5. In the **Target selection configuration** pane, choose **Off** for **Cross-zone load balancing**.
6. Choose **Save changes**.

AWS CLI

To turn off cross-zone load balancing

Use the modify-target-group-attributes command and set the `load_balancing.cross_zone.enabled` attribute to `false`.

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

CloudFormation

To turn off cross-zone load balancing

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: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "load_balancing.cross_zone.enabled"
          Value: "false"
```

Turn on cross-zone load balancing

You can turn on cross-zone load balancing for your Application Load Balancer target groups at any time. The cross-zone load balancing setting at the target group level overrides the setting at the load balancer level.

Console

To turn off cross-zone load balancing

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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, select **Edit**.
5. In the **Target selection configuration** pane, choose **On** for **Cross-zone load balancing**.
6. Choose **Save changes**.

AWS CLI

To turn on cross-zone load balancing

Use the modify-target-group-attributes command and set the `load_balancing.cross_zone.enabled` attribute to `true`.

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

CloudFormation

To turn on cross-zone load balancing

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: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "load_balancing.cross_zone.enabled"
          Value: "true"
```

Automatic Target Weights (ATW)

Automatic Target Weights (ATW) constantly monitors the targets running your applications, detecting significant performance deviations, known as anomalies. ATW provides the ability to dynamically adjust the amount of traffic routed to targets, through real time data anomaly detection.

Automatic Target Weights (ATW) performs anomaly detection on every Application Load Balancer in your account automatically. When anomalous targets are identified, ATW can automatically attempt to stabilize them by reducing the amount of traffic they're routed, known as anomaly mitigation. ATW continuously optimizes traffic distribution to maximize per-target success rates while minimizing target group failure rates.

Considerations:

- Anomaly detection currently monitors HTTP 5xx response codes coming from, and connection failures to, your targets. Anomaly detection is always on and can't be turned off.
- ATW is not supported when using Lambda as a target.

Contents

Anomaly detection

ATW anomaly detection monitors for any targets that are displaying a significant deviation in behavior from other targets in their target group. These deviations, called anomalies, are determined by comparing the percent errors of one target with the percent errors of other targets

in the target group. These errors can be both connection errors and HTTP error codes. Targets reporting significantly higher than their peers are then considered anomalous.

Anomaly detection requires a minimum of three healthy targets in the target group. When a target is registered to a target group it must pass the health checks before receiving traffic. After the target starts receiving traffic, ATW begins monitoring the target and continuously publishes the anomaly result. For targets without anomalies, the anomaly result is `normal`. For targets with anomalies, the anomaly result is `anomalous`.

ATW anomaly detection works independently from target group health checks. A target can be passing all target group health checks, but still be marked anomalous due to an elevated error rate. Targets becoming anomalous does not affect their target group health check status.

Anomaly detection status

You can view the current anomaly detection status. The following are the possible values:

- `normal` – No anomalies were detected.
- `anomalous` – Anomalies were detected.

Console

To view the anomaly detection status

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. Within the **Registered targets** table, the **Anomaly detection result** column displays the anomaly status of each target.

AWS CLI

To view the anomaly detection status

Use the describe-target-health command. The following example displays the status for every target in the specified target group.

```
aws elbv2 describe-target-health \
```

```
    --target-group-arn target-group-arn \
    --include AnomalyDetection
```

Anomaly mitigation

ATW anomaly mitigation routes traffic away from anomalous targets automatically, giving them an opportunity to recover.

Requirement

The anomaly mitigation function of ATW is only available when using the **Weighted random** routing algorithm.

During mitigation:

- ATW periodically adjusts the amount of traffic routed to anomalous targets. Currently, the period is every five seconds.
- ATW reduces the amount of traffic routed to anomalous targets to the minimum amount required to perform anomaly mitigation.
- Targets which are no longer detected as anomalous will gradually have more traffic routed to them until they reach parity with other normal targets in the target group.

Console

To turn on anomaly mitigation

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. In the **Traffic configuration** pane, verify that the selected value for **Load balancing algorithm** is **Weighted random**.

   When the weighted random algorithm is initially selected, anomaly detection is on by default.
6. Under **Anomaly mitigation**, ensure that **Turn on anomaly mitigation** is selected.
7. Choose **Save changes**.

AWS CLI

To turn on anomaly mitigation

Use the modify-target-group-attributes command with the `load_balancing.algorithm.anomaly_mitigation` attribute.

```
aws elbv2
```

Mitigation status

You can check whether ATW is performing mitigation on a target. The following are the possible values:

- `yes` – Mitigation is in progress.
- no – Mitigation is not in progress.

Console

To view the anomaly mitigation status

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. Within the **Registered targets** table, you can view the anomaly mitigation status of each target in the **Mitigation in effect** column.

AWS CLI

To view the anomaly mitigation status

Use the describe-target-health command. The following example displays the status for every target in the specified target group.

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

```
    --include AnomalyDetection
```

Sticky sessions

By default, an Application Load Balancer routes each request independently to a registered target based on the chosen load-balancing algorithm. However, you can use the sticky session feature (also known as session affinity) to enable the load balancer to bind a user's session to a specific target. This ensures that all requests from the user during the session are sent to the same target. This feature is useful for servers that maintain state information in order to provide a continuous experience to clients. To use sticky sessions, the client must support cookies.

Application Load Balancers support both duration-based cookies and application-based cookies. Sticky sessions are enabled at the target group level. You can use a combination of duration-based stickiness, application-based stickiness, and no stickiness across your target groups.

The key to managing sticky sessions is determining how long your load balancer should consistently route the user's request to the same target. If your application has its own session cookie, then you can use application-based stickiness and the load balancer session cookie follows the duration specified by the application's session cookie. If your application does not have its own session cookie, then you can use duration-based stickiness to generate a load balancer session cookie with a duration that you specify.

The content of load balancer generated cookies are encrypted using a rotating key. You can't decrypt or modify load balancer generated cookies.

For both stickiness types, the Application Load Balancer resets the expiry of the cookies it generates after every request. If a cookie expires, the session is no longer sticky and the client should remove the cookie from its cookie store.

Requirements

- An HTTP/HTTPS load balancer.
- At least one healthy instance in each Availability Zone.

Considerations

- Sticky sessions are not supported if cross-zone load balancing is disabled. Attempts to enable sticky sessions while cross-zone load balancing is disabled fail.

- For application-based cookies, cookie names have to be specified individually for each target group. However, for duration-based cookies, `AWSALB` is the only name used across all target groups.
- If you are using multiple layers of Application Load Balancers, you can enable sticky sessions across all layers with application-based cookies. However, with duration-based cookies, you can enable sticky sessions only on one layer, because `AWSALB` is the only name available.
- If the Application Load Balancer receives both an `AWSALBCORS` and an `AWSALB` duration-based stickiness cookie, the value in `AWSALBCORS` will take precedence.
- Application-based stickiness does not work with weighted target groups.
- If you have a forward action with multiple target groups, and sticky sessions are enabled for one or more of the target groups, you must enable stickiness at the target group level.
- WebSocket connections are inherently sticky. If the client requests a connection upgrade to WebSockets, the target that returns an HTTP 101 status code to accept the connection upgrade is the target used in the WebSockets connection. After the WebSockets upgrade is complete, cookie-based stickiness is not used.
- Application Load Balancers use the `Expires` attribute in the cookie header instead of the `Max-Age` attribute.
- Application Load Balancers do not support cookie values that are URL encoded.
- If the Application Load Balancer receives a new request while the target is draining due to deregistration, the request is routed to a healthy target.
- Sticky sessions are not supported if target optimizer is enabled.

Stickiness types

- Duration-based stickiness
- Application-based stickiness

Duration-based stickiness

Duration-based stickiness routes requests to the same target in a target group using a load balancer generated cookie (`AWSALB`). The cookie is used to map the session to the target. If your application does not have its own session cookie, you can specify your own stickiness duration and manage how long your load balancer should consistently route the user's request to the same target.

When a load balancer first receives a request from a client, it routes the request to a target (based on the chosen algorithm), and generates a cookie named `AWSALB`. It encodes information about the selected target, encrypts the cookie, and includes the cookie in the response to the client. The load balancer generated cookie has its own expiry of 7 days which is non-configurable.

In subsequent requests, the client should include the `AWSALB` cookie. When the load balancer receives a request from a client that contains the cookie, it detects it and routes the request to the same target. If the cookie is present but can't be decoded, or if it refers to a target that was deregistered or is unhealthy, the load balancer selects a new target and updates the cookie with information about the new target.

For cross-origin resource sharing (CORS) requests, some browsers require `SameSite=None; Secure` to enable stickiness. To support these browsers the load balancer always generates a second stickiness cookie, `AWSALBCORS`, which includes the same information as the original stickiness cookie, as well as the `SameSite` attribute. Clients receive both cookies, including non CORS requests.

Console

To enable duration-based stickiness

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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**, do the following:
   a. Select **Turn on stickiness**.
   b. For **Stickiness type**, select **Load balancer generated cookie**.
   c. For **Stickiness duration**, specify a value between 1 second and 7 days.
6. Choose **Save changes**.

AWS CLI

To enable duration-based stickiness

Use the modify-target-group-attributes command with the `stickiness.enabled` and `stickiness.lb_cookie.duration_seconds` attributes.

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes \
        "Key=stickiness.enabled,Value=true" \
        "Key=stickiness.lb_cookie.duration_seconds,Value=300"
```

CloudFormation

To enable duration-based stickiness

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the `stickiness.enabled` and `stickiness.lb_cookie.duration_seconds` attributes.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "stickiness.enabled"
          Value: "true"
        - Key: "stickiness.lb_cookie.duration_seconds"
          Value: "300"
```

Application-based stickiness

Application-based stickiness gives you the flexibility to set your own criteria for client-target stickiness. When you enable application-based stickiness, the load balancer routes the first request to a target within the target group based on the chosen algorithm. The target is expected to set a custom application cookie that matches the cookie configured on the load balancer to enable stickiness. This custom cookie can include any of the cookie attributes required by the application.

When the Application Load Balancer receives the custom application cookie from the target, it automatically generates a new encrypted application cookie to capture stickiness information. This load balancer generated application cookie captures stickiness information for each target group that has application-based stickiness enabled.

The load balancer generated application cookie does not copy the attributes of the custom cookie set by the target. It has its own expiry of 7 days which is non-configurable. In the response to the client, the Application Load Balancer only validates the name with which the custom cookie was configured at the target group level and not the value or the expiry attribute of the custom cookie. As long as the name matches, the load balancer sends both cookies, the custom cookie set by the target, and the application cookie generated by the load balancer, in the response to the client.

In subsequent requests, clients have to send back both cookies to maintain stickiness. The load balancer decrypts the application cookie, and checks whether the configured duration of stickiness is still valid. It then uses the information in the cookie to send the request to the same target within the target group to maintain stickiness. The load balancer also proxies the custom application cookie to the target without inspecting or modifying it. In subsequent responses, the expiry of the load balancer generated application cookie and the duration of stickiness configured on the load balancer are reset. To maintain stickiness between client and target, the expiry of the cookie, and the duration of stickiness should not elapse.

If a target fails or becomes unhealthy, the load balancer stops routing requests to that target, and chooses a new healthy target based on the chosen load balancing algorithm. The load balancer treats the session as now being "stuck" to the new healthy target, and continues routing requests to the new healthy target even if the failed target comes back.

With cross-origin resource sharing (CORS) requests, to enable stickiness, the load balancer adds the `SameSite=None; Secure` attributes to the load balancer generated application cookie only if the user-agent version is Chromium80 or above.

Because most browsers limit cookies to 4K in size, the load balancer shards application cookies greater than 4K into multiple cookies. Application Load Balancers support cookies up to 16K in size and can therefore create up to 4 shards that it sends to the client. The application cookie name that the client sees begins with “AWSALBAPP-" and includes a fragment number. For example, if the cookie size is 0-4K, the client sees AWSALBAPP-0. If the cookie size is 4-8k, the client sees AWSALBAPP-0 and AWSALBAPP-1, and so on.

Console

To enable application-based stickiness

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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**, do the following:
   a. Select **Turn on stickiness**.
   b. For **Stickiness type**, select **Application-based cookie**.
   c. For **Stickiness duration**, specify a value between 1 second and 7 days.
   d. For **App cookie name**, enter a name for your application-based cookie.

      Do not use `AWSALB`, `AWSALBAPP`, or `AWSALBTG` for the cookie name; they're reserved for use by the load balancer.
6. Choose **Save changes**.

AWS CLI

To enable application-based stickiness

Use the modify-target-group-attributes command with the following attributes:

- `stickiness.enabled`
- `stickiness.type`
- `stickiness.app_cookie.cookie_name`
- `stickiness.app_cookie.duration_seconds`

```
aws elbv2 modify-target-group-attributes \
    --target-group-arn target-group-arn \
    --attributes \
        "Key=stickiness.enabled,Value=true" \
        "Key=stickiness.type,Value=app_cookie" \
        "Key=stickiness.app_cookie.cookie_name,Value=my-cookie-name" \
        "Key=stickiness.app_cookie.duration_seconds,Value=300"
```

CloudFormation

To enable application-based stickiness

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource to include the following attributes:

- `stickiness.enabled`
- `stickiness.type`
- `stickiness.app_cookie.cookie_name`
- `stickiness.app_cookie.duration_seconds`

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      Protocol: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      TargetGroupAttributes:
        - Key: "stickiness.enabled"
          Value: "true"
        - Key: "stickiness.type"
          Value: "app_cookie"
        - Key: "stickiness.app_cookie.cookie_name"
          Value: "my-cookie-name"
        - Key: "stickiness.app_cookie.duration_seconds"
          Value: "300"
```

Manual rebalancing

When scaling up, if the number of targets increase considerably, there is potential for unequal distribution of load due to stickiness. In this scenario, you can rebalance the load on your targets using the following two options:

- Set an expiry on the cookie generated by the application that is prior to the current date and time. This prevents clients from sending the cookie to the Application Load Balancer, which will restart the process of establishing stickiness.
- Set a short duration on the load balancer's application-based stickiness configuration; for example, 1 second. This forces the Application Load Balancer to reestablish stickiness even if the cookie set by the target is not expired.

Register targets with your Application Load Balancer target group

You register your targets with a target group. When you create a target group, you specify its target type, which determines how you register its targets. For example, you can register instance IDs, IP addresses, or Lambda functions. For more information, see Target groups for your Application Load Balancers.

If demand on your currently registered targets increases, you can register additional targets in order to handle the demand. When your target is ready to handle requests, register it with your target group. The load balancer starts routing requests to the target as soon as the registration process completes and the target passes the initial health checks.

If demand on your registered targets decreases, or you need to service a target, you can deregister it from your target group. The load balancer stops routing requests to a target as soon as you deregister it. When the target is ready to receive requests, you can register it with the target group again.

When you deregister a target, the load balancer 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.

When you deregister a target that was registered by IP address, you must wait for the deregistration delay to complete before you can register the same IP address again.

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 target group 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*.

When shutting down an application on a target you must first deregister the target from its target group and allow time for existing connections to drain. You can monitor deregistration status using the describe-target-health CLI command, or by refreshing the target group view in the AWS Management Console. After confirming the target is deregistered you can proceed with stopping or terminating the application. This sequence prevents users from experiencing 5XX errors when applications are terminated while still processing traffic.

Target security groups

When you register EC2 instances as targets, you must ensure that the security groups for your instances allow the load balancer to communicate with your instances on both the listener port and the health check port.

Recommended rules

Inbound		
Source	**Port Range**	**Comment**
load balancer security group	*instance listener*	Allow traffic from the load balancer on the instance listener port
load balancer security group	*health check*	Allow traffic from the load balancer on the health check port

We also recommend that you allow inbound ICMP traffic to support Path MTU Discovery. For more information, see Path MTU Discovery in the *Amazon EC2 User Guide*.

Target Optimizer

Target optimizer lets you enforce strict concurrency on targets in a target group. It works with the help of an agent that you install and configure on targets. The agent serves as an inline proxy between the load balancer and your application. You configure the agent to enforce a maximum number of concurrent requests that the load balancer can send to the target. The agent tracks the number of requests the target is processing. When the number falls below the configured maximum value, the agent sends a signal to the load balancer letting it know that the target is ready to process another request.

To enable target optimizer, you specify a target control port when creating the target group. The load balancer establishes control channels with agents on this port for management traffic. This port is different from the port on which the load balancer sends application traffic. Targets registered with the target group must have the agent running on them.

Note: Target optimizer can only be enabled during target group creation. Target control port cannot be modified after creation.

The agent is available as a Docker image at: `public.ecr.aws/aws-elb/target-optimizer/target-control-agent:latest`. You configure the following environment variables when running the agent container:

`TARGET_CONTROL_DATA_ADDRESS`

The agent receives application traffic from the load balancer on this socket (IP:port). The port in this socket is the application traffic port you configure for the target group. By default, the agent can accept both plaintext and TLS connections.

`TARGET_CONTROL_CONTROL_ADDRESS`

The agent receives management traffic from the load balancer on this socket (IP:port). The port in the socket is the target control port you configure for the target group.

`TARGET_CONTROL_DESTINATION_ADDRESS`

The agent proxies application traffic to this socket (IP:port). Your application should be listening on this socket.

(Optional) `TARGET_CONTROL_MAX_CONCURRENCY`

The maximum number of concurrent requests that the target will receive from the load balancer. It can be between 0-1000. The default is 1.

(Optional) `TARGET_CONTROL_TLS_CERT_PATH`

The location of the TLS certificate that the agent provides to the load balancer during TLS handshake. By default, the agent generates a self-signed certificate in-memory.

(Optional) `TARGET_CONTROL_TLS_KEY_PATH`

The location of the private key corresponding to the TLS certificate that the agent provides to the load balancer during TLS handshake. By default, the agent generates a private key in-memory.

(Optional) `TARGET_CONTROL_TLS_SECURITY_POLICY`

The ELB security policy that you configure for the target group. The default is `ELBSecurityPolicy-2016-08`.

(Optional) `TARGET_CONTROL_PROTOCOL_VERSION`

The protocol through which the load balancer communicates with the agent. Possible values are `HTTP1`, `HTTP2`, `GRPC`. The default is `HTTP1`.

(Optional) `RUST_LOG`

The log level of the agent process. The agent software is written in Rust. Possible values are `debug`, `info`,and `error` . The default is `info`.

To modify the value for any environment variable, you have to restart the agent with the new value. You can monitor target optimizer with the following metrics: `TargetControlRequestCount`, `TargetControlRequestRejectCount`, `TargetControlActiveChannelCount`, `TargetControlNewChannelCount`, `TargetControlChannelErrorCount`, `TargetControlWorkQueueLength`, `TargetControlProcessedBytes`. For more information, see Target optimizer metrics For troubleshooting information, see Troubleshooting target optimizer

Shared subnets

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

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 how you register targets with that target group. For more information, see Target type.

Requirements and considerations

- An instance must be in the `running` state when you register it.
- A target instance must be in the virtual private cloud (VPC) that you specified for the target group.
- 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.
- You can't register the IP addresses of another Application Load Balancer in the same VPC. If the other Application Load Balancer is in a VPC that is peered to the load balancer VPC, you can register its IP addresses.

Console

To register targets

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 addresses and ports, and choose **Include as pending below**.
8. If the target type of the target group is `lambda`, select the Lambda function or enter its ARN. For more information, see Use Lambda functions 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 a Lambda function as a target.

```
aws elbv2 register-targets \
    --target-group-arn target-group-arn \
    --targets Id=lambda-function-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.

Console

To deregister targets

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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**.
6. When prompted for confirmation, 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 Lambda functions as targets of an Application Load Balancer

You can register your Lambda functions as targets and configure a listener rule to forward requests to the target group for your Lambda function. When the load balancer forwards the request to a

target group with a Lambda function as a target, it invokes your Lambda function and passes the content of the request to the Lambda function, in JSON format.

The load balancer invokes the Lambda function directly instead of using a network connection. Therefore, there are no requirements for the outbound rules of the Application Load Balancer security groups.

Limits

- The Lambda function and target group must be in the same account and in the same Region.
- The maximum size of the request body that you can send to a Lambda function is 1 MB. For related size limits, see HTTP header limits.
- The maximum size of the response JSON that the Lambda function can send is 1 MB.
- WebSockets are not supported. Upgrade requests are rejected with an HTTP 400 code.
- Local Zones are not supported.
- Automatic Target Weights (ATW) is not supported.

Contents

For a demo, see Lambda target on Application Load Balancer.

Prepare the Lambda function

The following recommendations apply if you are using your Lambda function with an Application Load Balancer.

Permissions to invoke the Lambda function

If you create the target group and register the Lambda function using the AWS Management Console, the console adds the required permissions to your Lambda function policy on your behalf. Otherwise, after you create the target group and register the function using the AWS CLI, you must use the add-permission command to grant ELB permission to invoke your Lambda function. We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to restrict function invocation to the specified target group. For more information, see The confused deputy problem in the *IAM User Guide*,

```
aws lambda add-permission \
    --function-name lambda-function-arn-with-alias-name \
    --statement-id elb1 \
    --principal elasticloadbalancing.amazonaws.com \
    --action lambda:InvokeFunction \
    --source-arn target-group-arn \
    --source-account target-group-account-id
```

Lambda function versioning

You can register one Lambda function per target group. To ensure that you can change your Lambda function and that the load balancer always invokes the current version of the Lambda function, create a function alias and include the alias in the function ARN when you register the Lambda function with the load balancer. For more information, see AWS Lambda function aliases in the *AWS Lambda Developer Guide*.

Function timeout

The load balancer waits until your Lambda function responds or times out. We recommend that you configure the timeout of the Lambda function based on your expected run time. For information about the default timeout value and how to change it, see Configure Lambda function timeout. For information about the maximum timeout value that you can configure, see AWS Lambda quotas.

Create a target group for the Lambda function

Create a target group, which is used in request routing. If the request content matches a listener rule with an action to forward it to this target group, the load balancer invokes the registered Lambda function.

Console

To create a target group and register the Lambda function

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/ec2/.
2. On the navigation pane, under **Load Balancing**, choose **Target Groups**.
3. Choose **Create target group**.
4. For **Choose a target type**, select **Lambda function**.
5. For **Target group name**, enter a name for the target group.
6. (Optional) To enable health checks, choose **Enable** in the **Health checks** section.
7. (Optional) Expand **Tags**. For each tag, choose **Add new tag** and enter a tag key and a tag value.
8. Choose **Next**.
9. If you are ready to register the Lambda function, choose **Select a Lambda function** and choose the Lambda function from the list, or choose **Enter a Lambda function ARN** and enter the ARN of the Lambda function,

   If you are not ready to register the Lambda function, choose **Register Lambda function later** and register the target later on. For more information, see the section called “Register targets”.
10. Choose **Create target group**.

AWS CLI

To create a target group of type lambda

Use the create-target-group command.

```
aws elbv2 create-target-group \
    --name my-target-group \
    --target-type lambda
```

To register the Lambda function

Use the register-targets command.

```
aws elbv2 register-targets \
```

```
    --target-group-arn target-group-arn \
    --targets Id=lambda-function-arn
```

CloudFormation

To create a target group and register the Lambda function

Define a resource of type AWS::ElasticLoadBalancingV2::TargetGroup. If you aren't ready to register the Lambda function now, you can omit the `Targets` property and add it later on.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      TargetType: lambda
      Tags:
        - Key: 'department'
          Value: '123'
      Targets:
        - Id: !Ref myLambdaFunction
```

Receive events from the load balancer

The load balancer supports Lambda invocation for requests over both HTTP and HTTPS. The load balancer sends an event in JSON format. The load balancer adds the following headers to every request: `X-Amzn-Trace-Id`, `X-Forwarded-For`, `X-Forwarded-Port`, and `X-Forwarded-Proto`.

If the `content-encoding` header is present, the load balancer Base64 encodes the body and sets `isBase64Encoded` to `true`.

If the `content-encoding` header is not present, Base64 encoding depends on the content type. For the following types, the load balancer sends the body as is and sets `isBase64Encoded` to `false`: text/*, application/json, application/javascript, and application/xml. Otherwise, the load balancer Base64 encodes the body and sets `isBase64Encoded` to `true`.

The following is an example event.

```
{
```

```
    "requestContext": {
        "elb": {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-target-
group/6d0ecf831eec9f09"
        }
    },
    "httpMethod": "GET",
    "path": "/",
    "queryStringParameters": {parameters},
    "headers": {
        "accept": "text/html,application/xhtml+xml",
        "accept-language": "en-US,en;q=0.8",
        "content-type": "text/plain",
        "cookie": "cookies",
        "host": "lambda-846800462-us-east-2.elb.amazonaws.com",
        "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)",
        "x-amzn-trace-id": "Root=1-5bdb40ca-556d8b0c50dc66f0511bf520",
        "x-forwarded-for": "72.21.198.66",
        "x-forwarded-port": "443",
        "x-forwarded-proto": "https"
    },
    "isBase64Encoded": false,
    "body": "request_body"
}
```

Respond to the load balancer

The response from your Lambda function must include the Base64 encoding status, status code, and headers. You can omit the body.

To include a binary content in the body of the response, you must Base64 encode the content and set `isBase64Encoded` to `true`. The load balancer decodes the content to retrieve the binary content and sends it to the client in the body of the HTTP response.

The load balancer does not honor hop-by-hop headers, such as `Connection` or `Transfer-Encoding`. You can omit the `Content-Length` header because the load balancer computes it before sending responses to clients.

The following is an example response from a **nodejs** based Lambda function.

```
{
```

```
    "isBase64Encoded": false,
    "statusCode": 200,
    "statusDescription": "200 OK",
    "headers": {
        "Set-cookie": "cookies",
        "Content-Type": "application/json"
    },
    "body": "Hello from Lambda (optional)"
}
```

For Lambda function templates that work with Application Load Balancers, see application-load-balancer-serverless-app on github. Alternatively, open the Lambda console, choose **Applications**, **Create a application**, and select one of the following from the AWS Serverless Application Repository:

- ALB-Lambda-Target-UploadFiletoS3
- ALB-Lambda-Target-BinaryResponse
- ALB-Lambda-Target-WhatisMyIP

Multi-value headers

If requests from a client or responses from a Lambda function contain headers with multiple values or contains the same header multiple times, or query parameters with multiple values for the same key, you can enable support for multi-value header syntax. After you enable multi-value headers, the headers and query parameters exchanged between the load balancer and the Lambda function use arrays instead of strings. If you do not enable multi-value header syntax and a header or query parameter has multiple values, the load balancer uses the last value that it receives.

Contents

Requests with multi-value headers

The names of the fields used for headers and query string parameters differ depending on whether you enable multi-value headers for the target group.

The following example request has two query parameters with the same key:

```
http://www.example.com?&myKey=val1&myKey=val2
```

With the default format, the load balancer uses the last value sent by the client and sends you an event that includes query string parameters using `queryStringParameters`. For example:

```
"queryStringParameters": { "myKey": "val2"},
```

If you enable multi-value headers, the load balancer uses both key values sent by the client and sends you an event that includes query string parameters using `multiValueQueryStringParameters`. For example:

```
"multiValueQueryStringParameters": { "myKey": ["val1", "val2"] },
```

Similarly, suppose that the client sends a request with two cookies in the header:

```
"cookie": "name1=value1",
"cookie": "name2=value2",
```

With the default format, the load balancer uses the last cookie sent by the client and sends you an event that includes headers using `headers`. For example:

```
"headers": {
    "cookie": "name2=value2",
    ...
},
```

If you enable multi-value headers, the load balancer uses both cookies sent by the client and sends you an event that includes headers using `multiValueHeaders`. For example:

```
"multiValueHeaders": {
    "cookie": ["name1=value1", "name2=value2"],
    ...
},
```

If the query parameters are URL-encoded, the load balancer does not decode them. You must decode them in your Lambda function.

Responses with multi-value headers

The names of the fields used for headers differ depending on whether you enable multi-value headers for the target group. You must use `multiValueHeaders` if you have enabled multi-value headers and `headers` otherwise.

With the default format, you can specify a single cookie:

```
{
  "headers": {
      "Set-cookie": "cookie-name=cookie-value;Domain=myweb.com;Secure;HttpOnly",
      "Content-Type": "application/json"
  },
}
```

If you enable multi-value headers, you must specify multiple cookies as follows:

```
{
  "multiValueHeaders": {
      "Set-cookie": ["cookie-name=cookie-
value;Domain=myweb.com;Secure;HttpOnly","cookie-name=cookie-value;Expires=May 8,
 2019"],
      "Content-Type": ["application/json"]
  },
}
```

The load balancer might send the headers to the client in a different order than the order specified in the Lambda response payload. Therefore, do not count on headers being returned in a specific order.

Enable multi-value headers

You can enable or disable multi-value headers for a target group with the target type `lambda`.

Console

To enable multi-value headers

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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. Enable **Multi value headers**.
6. Choose **Save changes**.

AWS CLI

To enable multi-value headers

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

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

CloudFormation

To enable multi-value headers

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

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      TargetType: lambda
      Tags:
        - Key: 'department'
          Value: '123'
      Targets:
        - Id: !Ref myLambdaFunction
      TargetGroupAttributes:
        - Key: "lambda.multi_value_headers.enabled"
          Value: "true"
```

Enable health checks

By default, health checks are disabled for target groups of type `lambda`. You can enable health checks in order to implement DNS failover with Amazon Route 53. The Lambda function can check the health of a downstream service before responding to the health check request. If the response from the Lambda function indicates a health check failure, the health check failure is passed to Route 53. You can configure Route 53 to fail over to a backup application stack.

You are charged for health checks as you are for any Lambda function invocation.

The following is the format of the health check event sent to your Lambda function. To check whether an event is a health check event, check the value of the user-agent field. The user agent for health checks is `ELB-HealthChecker/2.0`.

```
{
    "requestContext": {
        "elb": {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-target-
group/6d0ecf831eec9f09"
        }
    },
    "httpMethod": "GET",
    "path": "/",
    "queryStringParameters": {},
    "headers": {
        "user-agent": "ELB-HealthChecker/2.0"
    },
    "body": "",
    "isBase64Encoded": false
}
```

Console

To enable health checks for a lambda target group

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 **Health checks** tab, choose **Edit**.

5. For **Health checks**, select **Enable**.
6. (Optional) Update the health check settings as needed.
7. Choose **Save changes**.

AWS CLI

To enable health checks for a lambda target group

Use the modify-target-group command.

```
aws elbv2 modify-target-group \
    --target-group-arn target-group-arn \
    --health-check-enabled
```

CloudFormation

To enable health checks for a lambda target group

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      TargetType: lambda
      HealthCheckEnabled: true
      Tags:
        - Key: 'department'
          Value: '123'
      Targets:
        - Id: !Ref myLambdaFunction
```

Register the Lambda function

You can register a single Lambda function with each target group. To replace a Lambda function, we recommend that you create a new target group, register the new function with the new target group, and update the listener rules to use the new target group.

Console

To register a Lambda function

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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, if there is no Lambda function registered, choose **Register target**.
5. Select the Lambda function or enter its ARN.
6. Choose **Register**.

AWS CLI

To register a Lambda function

Use the register-targets command.

```
aws elbv2 register-targets \
    --target-group-arn target-group-arn \
    --targets Id=lambda-function-arn
```

CloudFormation

To register a Lambda function

Update the AWS::ElasticLoadBalancingV2::TargetGroup resource.

```
Resources:
  myTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: my-target-group
      TargetType: lambda
      Tags:
        - Key: 'department'
          Value: '123'
      Targets:
        - Id: !Ref myLambdaFunction
```

Deregister the Lambda function

If you no longer need to send traffic to your Lambda function, you can deregister it. After you deregister a Lambda function, in-flight requests fail with HTTP 5XX errors.

To replace a Lambda function, we recommend that you create a new target group, register the new function with the new target group, and update the listener rules to use the new target group.

Console

To deregister a Lambda function

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 target and choose **Deregister**.
5. When prompted for confirmation, choose **Deregister**.

AWS CLI

To deregister a Lambda function

Use the deregister-targets command.

```
aws elbv2 deregister-targets \
    --target-group-arn target-group-arn \
    --targets Id=lambda-function-arn
```

Tags for your Application Load Balancer target group

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

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

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

Restrictions

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case-sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

Console

To manage the tags for a target group

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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: HTTP
      Port: 80
      TargetType: ip
      VpcId: !Ref myVPC
      Tags:
        - Key: 'project'
          Value: 'lima'
        - Key: 'department'
          Value: 'digital-media'
```

Delete an Application Load Balancer target group

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

Console

To delete a target group

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 Application 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 Application Load Balancer.

Access logs

You can use access logs to capture detailed information about the requests made to your load balancer and store them as log files 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 Application Load Balancer.

Connection logs

You can use connection logs to capture attributes about the requests sent to your load balancer, and store them as log files in Amazon S3. You can use these connection logs to determine the client IP address and port, client certificate information, connection results, and TLS ciphers being used. These connection logs can then be used to review request patterns, and other trends. For more information, see Connection logs for your Application Load Balancer.

Health check logs

You can use health check logs to capture detailed information about the health checks made to your registered targets for your load balancer and store them as log files in Amazon S3. You can use these health check logs to troubleshoot issues with your targets. For more information, see Health check logs.

Request tracing

You can use request tracing to track HTTP requests. The load balancer adds a header with a trace identifier to each request it receives. For more information, see Request tracing for your Application Load Balancer.

CloudTrail logs

You can use AWS CloudTrail to capture detailed information about the calls made to the ELB 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 ELB using CloudTrail.

CloudWatch metrics for your Application Load Balancer

ELB 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.

ELB reports metrics to CloudWatch only when requests are flowing through the load balancer. If there are requests flowing through the load balancer, ELB 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.

Metrics for Application Load Balancers exclude health check requests.

For more information, see the Amazon CloudWatch User Guide.

Contents

Application Load Balancer metrics

- Load balancers
- LCUs
- Targets
- Target group health
- Lambda functions
- User authentication
- Target Optimizer

The AWS/ApplicationELB namespace includes the following metrics for load balancers.

Metric	Description
ActiveConnectionCount	The total number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
BYoIPUtilPercentage	The percentage of usage from the IP pool. **Reporting criteria**: BYoIP is enabled on the load balancer. **Statistics**: The only meaningful statistic is Average. **Dimensions** • LoadBalancer ,TargetGroup • LoadBalancer ,TargetGroup ,AvailabilityZone

Metric	Description
ClientTLSNegotiationErrorCount	The number of TLS connections initiated by the client that did not establish a session with the load balancer due to a TLS error. Possible causes include a mismatch of ciphers or protocols or the client failing to verify the server certificate and closing the connection. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
DesyncMitigationMode_NonCompliant_Request_Count	The number of requests that do not comply with RFC 7230. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
GrpcRequestCount	The number of gRPC requests processed over IPv4 and IPv6. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1. **Dimensions** • LoadBalancer ,TargetGroup • AvailabilityZone ,LoadBalancer ,TargetGroup • TargetGroup • AvailabilityZone ,TargetGroup
HTTP_Fixed_Response_Count	The number of fixed-response actions that were successful. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
HTTP_Redirect_Count	The number of redirect actions that were successful. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
HTTP_Redirect_Url_Limit_Exceeded_Count	The number of redirect actions that couldn't be completed because the URL in the response location header is larger than 8K. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
HTTPCode_ELB_3XX_Count	The number of HTTP 3XX redirection codes that originate from the load balancer. This count does not include response codes generated by targets. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
HTTPCode_ELB_4XX_Count	The number of HTTP 4XX client error codes that originate from the load balancer. This count does not include response codes generated by targets. Client errors are generated when requests are malformed or incomplete. These requests were not received by the target, other than in the case where the load balancer returns an HTTP 460 error code. This count does not include any response codes generated by the targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. `Minimum`, `Maximum`, and `Average` all return 1. **Dimensions** • `LoadBalancer` • `AvailabilityZone ,LoadBalancer`
HTTPCode_ELB_5XX_Count	The number of HTTP 5XX server error codes that originate from the load balancer. This count does not include any response codes generated by the targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. `Minimum`, `Maximum`, and `Average` all return 1. **Dimensions** • `LoadBalancer` • `AvailabilityZone ,LoadBalancer`

Metric	Description
HTTPCode_ELB_500_Count	The number of HTTP 500 error codes that originate from the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
HTTPCode_ELB_502_Count	The number of HTTP 502 error codes that originate from the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
HTTPCode_ELB_503_Count	The number of HTTP 503 error codes that originate from the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
HTTPCode_ELB_504_Count	The number of HTTP 504 error codes that originate from the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
IPv6ProcessedBytes	The total number of bytes processed by the load balancer over IPv6. This count is included in ProcessedBytes . **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
IPv6RequestCount	The number of IPv6 requests received by the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. Minimum, Maximum, and Average all return 1. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
LowReputationPacketsDropped	The number of packets dropped from known malicious sources. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AavailabilityZone ,LoadBalancer
LowReputationRequestsDenied	The number of HTTP requests denied with an HTTP 403 response. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
NewConnectionCount	The total number of new TCP connections established from clients to the load balancer and from the load balancer to targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
NonStickyRequestCount	The number of requests where the load balancer chose a new target because it couldn't use an existing sticky session. For example, the request was the first request from a new client and no stickiness cookie was presented, a stickiness cookie was presented but it did not specify a target that was registered with this target group, the stickiness cookie was malformed or expired, or an internal error prevented the load balancer from reading the stickiness cookie. **Reporting criteria**: Stickiness is enabled on the target group. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
ProcessedBytes	The total number of bytes processed by the load balancer over IPv4 and IPv6 (HTTP header and HTTP payload). This count includes traffic to and from clients and Lambda functions, traffic over Websocket connections, and traffic from an Identity Provider (IdP) if user authentication is enabled. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
RejectedConnectionCount	The number of connections that were rejected because the load balancer had reached its maximum number of connections. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
RequestCount	The number of requests processed over IPv4 and IPv6. This metric is only incremented for requests where the load balancer node was able to choose a target. Requests that are rejected before a target is chosen are not reflected in this metric. **Reporting criteria**: Reported if there are registered targets. **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • LoadBalancer ,AvailabilityZone • LoadBalancer ,TargetGroup • LoadBalancer ,AvailabilityZone ,TargetGroup

Metric	Description
RuleEvaluations	The number of rules evaluated by the load balancer while processin g requests. The default rule is not counted. The 10 free rule evaluations per request are included in this count. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for load balancer capacity units (LCU).

Metric	Description
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. When LCU reservation is active, ConsumedLCUs will report 0 if usage is below the reserved capacity, and will report values above 0 if usage exceeds the reserved LCUs. For more information, see Elastic Load Balancing pricing. **Reporting criteria**: Always reported **Statistics**: All **Dimensions** • LoadBalancer
PeakLCUs	The maximum number of load balancer capacity units (LCU) used by your load balancer at a given point in time. Only applicable when using LCU Reservation. **Reporting criteria**: Always

Metric	Description
	Statistics: The most useful statistics are `Sum` and `Max`. **Dimensions** • `LoadBalancer`
`ReservedLCUs`	A billing metric that reports the reserved capacity on a per-minut e basis. The total ReservedLCUs over any period is the amount of LCUs you will be charged for. For example, if 500 LCUs are reserved for an hour, the per-minute metric will be 8.33 LCUs. For more information, see Monitor reservation. **Reporting criteria**: There is a nonzero value **Statistics**: All **Dimensions** • `LoadBalancer`

The `AWS/ApplicationELB` namespace includes the following metrics for targets.

Metric	Description
`AnomalousHostCount`	The number of hosts detected with anomalies. **Reporting criteria**: Always reported **Statistics**: The only meaningful statistics are `Minimum` and `Maximum`. **Dimensions** • `TargetGroup ,LoadBalancer` • `TargetGroup ,AvailabilityZone ,LoadBalancer`
`HealthyHostCount`	The number of targets that are considered healthy.

Metric	Description
	Reporting criteria: Reported if there are registered targets. **Statistics**: The most useful statistics are `Average`, `Minimum`, and `Maximum`. **Dimensions** • `LoadBalancer ,TargetGroup` • `LoadBalancer ,AvailabilityZone ,TargetGroup`
`HTTPCode_Target_2XX_Count ,HTTPCode_Target_3XX_Count ,HTTPCode_Target_4XX_Count ,HTTPCode_Target_5XX_Count`	The number of HTTP response codes generated by the targets. This does not include any response codes generated by the load balancer. **Reporting criteria**: Reported if there are registered targets. **Statistics**: The most useful statistic is Sum. `Minimum`, `Maximum`, and `Average` all return 1. **Dimensions** • `LoadBalancer` • `AvailabilityZone ,LoadBalancer` • `TargetGroup ,LoadBalancer` • `TargetGroup ,AvailabilityZone ,LoadBalancer`
`MitigatedHostCount`	The number of targets under mitigation. **Reporting criteria**: Always reported **Statistics**: The most useful statistics are `Average`, `Minimum`, and `Maximum`. **Dimensions** • `TargetGroup ,LoadBalancer` • `TargetGroup ,AvailabilityZone ,LoadBalancer`

Metric	Description
RequestCountPerTarget	The average request count per target, in a target group. You must specify the target group using the TargetGroup dimension. This metric does not apply if the target is a Lambda function. This count uses the total number of requests received by the target group, divided by the number of healthy targets in the target group. If there are no healthy targets in the target group, it is divided by the total number of registered targets. **Reporting criteria**: Always reported **Statistics**: The only valid statistic is Sum. This represents the average not the sum. **Dimensions** • TargetGroup • TargetGroup ,AvailabilityZone • LoadBalancer ,TargetGroup • LoadBalancer ,AvailabilityZone ,TargetGroup
TargetConnectionErrorCount	The number of connections that were not successfully established between the load balancer and target. This metric does not apply if the target is a Lambda function. This metric is not incremented for unsuccessful health check connections. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer • TargetGroup ,LoadBalancer • TargetGroup ,AvailabilityZone ,LoadBalancer

Metric	Description
TargetResponseTime	The time elapsed, in seconds, after the request leaves the load balancer until the target starts to send the response headers. This is equivalent to the target_processing_time field in the access logs. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistics are Average and pNN.NN (percentiles). **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer • TargetGroup ,LoadBalancer • TargetGroup ,AvailabilityZone ,LoadBalancer
TargetTLSNegotiationErrorCount	The number of TLS connections initiated by the load balancer that did not establish a session with the target. Possible causes include a mismatch of ciphers or protocols. This metric does not apply if the target is a Lambda function. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer • TargetGroup ,LoadBalancer • TargetGroup ,AvailabilityZone ,LoadBalancer

Metric	Description
UnHealthyHostCount	The number of targets that are considered unhealthy. When you deregister a target, this decreases HealthyHostCount but does not increase UnhealthyHostCount . **Reporting criteria**: Reported if there are registered targets. **Statistics**: The most useful statistics are Average, Minimum, and Maximum. **Dimensions** • LoadBalancer , TargetGroup • LoadBalancer , AvailabilityZone , TargetGroup
ZonalShiftedHostCount	The number of targets that are considered disabled due to zonal shift. **Reporting criteria**: Reported when there is a value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer , TargetGroup . • AvailabilityZone , LoadBalancer , TargetGroup .

The AWS/ApplicationELB namespace includes the following metrics for target group health. For more information, see the section called “Target group health”.

Metric	Description
HealthyStateDNS	The number of zones that meet the DNS healthy state requirements. **Statistics**: The most useful statistic is Max.

Metric	Description
	Dimensions • LoadBalancer ,TargetGroup • AvailabilityZone ,LoadBalancer ,TargetGroup
HealthyStateRouting	The number of zones that meet the routing healthy state requirements. **Statistics**: The most useful statistic is Max. **Dimensions** • LoadBalancer ,TargetGroup • AvailabilityZone ,LoadBalancer ,TargetGroup
UnhealthyRoutingRequestCount	The number of requests that are routed using the routing failover action (fail open). **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer ,TargetGroup • AvailabilityZone ,LoadBalancer ,TargetGroup
UnhealthyStateDNS	The number of zones that do not meet the DNS healthy state requirements and therefore were marked unhealthy in DNS. **Statistics**: The most useful statistic is Min. **Dimensions** • LoadBalancer ,TargetGroup • AvailabilityZone ,LoadBalancer ,TargetGroup

Metric	Description
UnhealthyStateRouting	The number of zones that do not meet the routing healthy state requirements, and therefore the load balancer distributes traffic to all targets in the zone, including the unhealthy targets. **Statistics**: The most useful statistic is Min. **Dimensions** • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup

The AWS/ApplicationELB namespace includes the following metrics for Lambda functions that are registered as targets.

Metric	Description
LambdaInternalError	The number of requests to a Lambda function that failed because of an issue internal to the load balancer or AWS Lambda. To get the error reason codes, check the error_reason field of the access log. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • TargetGroup • TargetGroup , LoadBalancer
LambdaTargetProcessedBytes	The total number of bytes processed by the load balancer for requests to and responses from a Lambda function. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum.

Metric	Description
	Dimensions • LoadBalancer
LambdaUserError	The number of requests to a Lambda function that failed because of an issue with the Lambda function. For example, the load balancer did not have permission to invoke the function, the load balancer received JSON from the function that is malformed or missing required fields, or the size of the request body or response exceeded the maximum size of 1 MB. To get the error reason codes, check the error_reason field of the access log. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • TargetGroup • TargetGroup ,LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for user authentication.

Metric	Description
ELBAuthError	The number of user authentications that could not be completed because an authenticate action was misconfigured, the load balancer couldn't establish a connection with the IdP, or the load balancer couldn't complete the authentication flow due to an internal error. To get the error reason codes, check the error_reason field of the access log. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum.

Metric	Description
	Dimensions • LoadBalancer • AvailabilityZone ,LoadBalancer
ELBAuthFailure	The number of user authentications that could not be completed because the IdP denied access to the user or an authorization code was used more than once. To get the error reason codes, check the error_reason field of the access log. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
ELBAuthLatency	The time elapsed, in milliseconds, to query the IdP for the ID token and user info. If one or more of these operations fail, this is the time to failure. **Reporting criteria**: There is a nonzero value **Statistics**: All statistics are meaningful. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
ELBAuthRefreshTokenSuccess	The number of times the load balancer successfully refreshed user claims using a refresh token provided by the IdP. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
ELBAuthSuccess	The number of authenticate actions that were successful. This metric is incremented at the end of the authentication workflow, after the load balancer has retrieved the user claims from the IdP. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
ELBAuthUserClaimsSizeExceeded	The number of times that a configured IdP returned user claims that exceeded 11K bytes in size. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for target optimizer.

Metric	Description
TargetControlRequestCount	Number of requests forwarded by ALB to agents. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
TargetControlRequestRejectCount	Number of requests rejected by ALB due to no targets being ready to receive requests. This metric shows an uptick when TargetControlWorkQueueLength is zero. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
TargetControlActiveChannelCount	Number of active control channels between ALB and agents. For a load balancer, this should be equal to the number of agents. A lower than expected number indicates that agents are not configured properly or are not available. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum.

Metric	Description
	Dimensions • LoadBalancer • AvailabilityZone ,LoadBalancer
TargetControlNewChannelCount	Number of new control channels created between ALB and agents. You will see an uptick in this metric when a new target with the agent installed is successfully added to the target group. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
TargetControlChannelErrorCount	Number of control channels between ALB and agents that failed to establish or experienced an unexpected error. A control channel error will result in that agent (and target) not receiving any application traffic. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric	Description
TargetControlWorkQueueLength	Number of signals received by the ALB from agents asking for requests. This data comes from snapshots taken at 1-minute intervals. Sub-minute changes are not captured. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The only meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer
TargetControlProcessedBytes	Number of bytes processed by ALB for traffic to target groups that enable target optimizer. **Reporting criteria**: Target optimizer is enabled on a target group and there is a nonzero value. **Statistics**: The most meaningful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone ,LoadBalancer

Metric dimensions for Application Load Balancers

To filter the metrics for your Application 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: app/*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 Application Load Balancer metrics

CloudWatch provides statistics based on the metric data points published by ELB. 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. For example, suppose there are 2 load balancer nodes that make up the Application Load Balancer. One node has HealthyHostCount with a Minimum of 2, a Maximum of 10, and an Average of 6, while the other node has HealthyHostCount with a Minimum of 1, a Maximum of 5, and an Average of 3. Therefore, the load balancer has a Minimum of 1, a Maximum of 10, and an Average of about 4.

We recommend you monitor for non-zero UnHealthyHostCount in the Minimum statistic, and alarm on non-zero value for more than one data point. Using the Minimum will detect when targets are considered unhealthy by every node and Availability Zone of your load balancer. Alarming on Average or Maximum is useful if you want to be alerted to potential problems, and we recommend customers review this metric and investigate non-zero occurrences. Mitigating failures automatically can be done following best practices of using load balancer health check in Amazon EC2 Auto Scaling, or Amazon Elastic Container Service (Amazon ECS).

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.

A percentile indicates the relative standing of a value in a data set. You can specify any percentile, using up to two decimal places (for example, p95.45). For example, the 95th percentile means that 95 percent of the data is below this value and 5 percent is above. Percentiles are often used to isolate anomalies. For example, suppose that an application serves the majority of requests from a cache in 1-2 ms, but in 100-200 ms if the cache is empty. The maximum reflects the slowest case, around 200 ms. The average doesn't indicate the distribution of the data. Percentiles provide a more meaningful view of the application's performance. By using the 99th percentile as an Auto Scaling trigger or a CloudWatch alarm, you can target that no more than 1 percent of requests take longer than 2 ms to process.

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://eusc-de-east-1.console.amazonaws-eusc.eu/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 then choose the **Monitoring** tab.
   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 then choose the **Monitoring** tab.

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://eusc-de-east-1.console.amazonaws-eusc.eu/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **ApplicationELB** namespace.
4. (Optional) To view a metric across all dimensions, enter its name in the search field.
5. (Optional) To filter by dimension, select one of the following:

   - To display only the metrics reported for your load balancers, choose **Per AppELB Metrics**. To view the metrics for a single load balancer, enter its name in the search field.
   - To display only the metrics reported for your target groups, choose **Per AppELB, per TG Metrics**. To view the metrics for a single target group, enter its name in the search field.
   - To display only the metrics reported for your load balancers by Availability Zone, choose **Per AppELB, per AZ Metrics**. To view the metrics for a single load balancer, enter its name in the search field. To view the metrics for a single Availability Zone, enter its name in the search field.
   - To display only the metrics reported for your load balancers by Availability Zone and target group, choose **Per AppELB, per AZ, per TG Metrics**. To view the metrics for a single load balancer, enter its name in the search field. To view the metrics for a single target group, enter its name in the search field. To view the metrics for a single Availability Zone, enter 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/ApplicationELB
```

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. 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/ApplicationELB \
--metric-name UnHealthyHostCount --statistics Average  --period 3600 \
--dimensions Name=LoadBalancer,Value=app/my-load-balancer/50dc6c495c0c9188 \
Name=TargetGroup,Value=targetgroup/my-targets/73e2d6bc24d8a067 \
--start-time 2016-04-18T00:00:00Z --end-time 2016-04-21T00:00:00Z
```

The following is example output:

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-04-18T22:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2016-04-18T04:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        ...
    ],
    "Label": "UnHealthyHostCount"
}
```

Access logs for your Application Load Balancer

ELB provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues.

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

You are charged storage costs for Amazon S3, but not charged for the bandwidth used by ELB to send log files to Amazon S3. For more information about storage costs, see Amazon S3 pricing.

Contents

Access log files

ELB 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_app.load-balancer-id_end-time_ip-address_random-string.log.gz
```

bucket

The name of the S3 bucket.

prefix

(Optional) The prefix (logical hierarchy) for the bucket. The prefix that you specify must not include the string `AWSLogs`. For more information, see Organizing objects using prefixes.

`AWSLogs`

We add the portion of the file name starting with `AWSLogs` after the bucket name and optional prefix that you specify.

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 20140215T2340Z contains entries for requests made between 23:35 and 23:40 in UTC or Zulu time.

ip-address

The IP address of the load balancer node that handled the request. For an internal load balancer, this is a private IP address.

random-string

A system-generated random string.

The following is an example log file name with a prefix:

```
s3://amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/
elasticloadbalancing/us-east-2/2022/05/01/123456789012_elasticloadbalancing_us-
east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

The following is an example log file name without a prefix:

```
s3://amzn-s3-demo-logging-bucket/AWSLogs/123456789012/elasticloadbalancing/
us-east-2/2022/05/01/123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_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 Object lifecycle management in the *Amazon S3 User Guide*.

Access log entries

ELB logs requests sent to the load balancer, including requests that never made it to the targets. For example, if a client sends a malformed request, or there are no healthy targets to respond to the request, the request is still logged.

Each log entry contains the details of a single request (or connection in the case of WebSockets) made to the load balancer. For WebSockets, an entry is written only after the connection is closed. If the upgraded connection can't be established, the entry is the same as for an HTTP or HTTPS request.

> **Important**
>
> ELB logs 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.

Contents

Syntax

The following table describes the fields of an access log entry, in order. All fields are delimited by spaces. When we add a new field, we add it to the end of the log entry. As we prepare to release a new field, you might see an additional trailing "-" before the field is released. Ensure that you configure log parsing to stop after the last documented field, and update log parsing after we release a new field.

Field (position)	Description
type (1)	The type of request or connection. The possible values are as follows (ignore any other values): • `http` — HTTP • `https` — HTTP over TLS • `h2` — HTTP/2 over TLS • `grpcs`— gRPC over TLS • `ws` — WebSockets • `wss` — WebSockets over TLS
time (2)	The time when the load balancer generated a response to the client, in ISO 8601 format. For WebSockets, this is the time when the connection is closed.
elb (3)	The resource ID of the load balancer. If you are parsing access log entries, note that resources IDs can contain forward slashes (/).
client:port (4)	The IP address and port of the requesting client. If there is a proxy in front of the load balancer, this field contains the IP address of the proxy.
target:port (5)	The IP address and port of the target that processed this request. If the client didn't send a full request, the load balancer can't dispatch the request to a target, and this value is set to -. If the target is a Lambda function, this value is set to -. If the request is blocked by AWS WAF, this value is set to -.
request_processing_time (6)	The total time elapsed (in seconds, with millisecond precision) from the time the load balancer received the request until the time it sent the request to a target.

Field (position)	Description
	This value is set to -1 if the load balancer can't dispatch the request to a target. This can happen if the target closes the connection before the idle timeout or if the client sends a malformed request. This value can also be set to -1 if a TCP connection cannot be established with the target before reaching the 10-second TCP connection timeout. If AWS WAF is enabled for your Application Load Balancer or the target type is a Lambda function, the time it takes for the client to send the required data for POST requests is counted towards `request_processing_time` .
target_processing_time (7)	The total time elapsed (in seconds, with millisecond precision) from the time the load balancer sent the request to a target until the target started to send the response headers. This value is set to -1 if the load balancer can't dispatch the request to a target. This can happen if the target closes the connection before the idle timeout or if the client sends a malformed request. This value can also be set to -1 if the registered target does not respond before the idle timeout. If AWS WAF is not enabled for your Application Load Balancer, the time it takes for the client to send the required data for POST requests is counted towards `target_processing_time` .
response_processing_time (8)	The total time elapsed (in seconds, with millisecond precision) from the time the load balancer received the response header from the target until it started to send the response to the client. This includes both the queuing time at the load balancer and the connection acquisition time from the load balancer to the client. This value is set to -1 if the load balancer doesn't receive a response from a target. This can happen if the target closes the connection before the idle timeout or if the client sends a malformed request.

Field (position)	Description
elb_status_code (9)	The status code of the response generated by the load balancer, fixed response rule, or AWS WAF custom response code for Block actions.
target_status_code (10)	The status code of the response from the target. This value is recorded only if a connection was established to the target and the target sent a response. Otherwise, it is set to -.
received_bytes (11)	The size of the request, in bytes, received from the client (requester). For HTTP requests, this includes the headers. For WebSockets, this is the total number of bytes received from the client on the connection.
sent_bytes (12)	The size of the response, in bytes, sent to the client (requester). For HTTP requests, this includes the response headers and body. For WebSockets, this is the total number of bytes sent to the client on the connection. The TCP headers and TLS handshake payload are not included in `sent_bytes` . Therefore `sent_bytes` won't match `DataTransfer-Out-Bytes` in AWS Cost Explorer.
"request_line" (13)	The request line from the client, enclosed in double quotes and logged using the following format: HTTP method + protocol://host:port/uri + HTTP version. The load balancer preserves the URL sent by the client, as is, when recording the request URI. It does not set the content type for the access log file. When you process this field, consider how the client sent the URL.
"user_agent" (14)	A User-Agent string that identifies the client that originated the request, enclosed in double quotes. The string consists of one or more product identifiers, product[/version]. If the string is longer than 8 KB, it is truncated.
ssl_cipher (15)	[HTTPS listener] The SSL cipher. This value is set to - if the listener is not an HTTPS listener.
ssl_protocol (16)	[HTTPS listener] The SSL protocol. This value is set to - if the listener is not an HTTPS listener.

Field (position)	Description
target_group_arn (17)	The Amazon Resource Name (ARN) of the target group.
"trace_id" (18)	The contents of the **X-Amzn-Trace-Id** header, enclosed in double quotes.
"domain_name" (19)	[HTTPS listener] The SNI domain provided by the client during the TLS handshake, enclosed in double quotes. This value is set to - if the client doesn't support SNI or the domain doesn't match a certificate and the default certificate is presented to the client.
"chosen_cert_arn" (20)	[HTTPS listener] The ARN of the certificate presented to the client, enclosed in double quotes. This value is set to `session-reused` if the session is reused. This value is set to - if the listener is not an HTTPS listener.
matched_rule_priority (21)	The priority value of the rule that matched the request. If a rule matched, this is a value from 1 to 50,000. If no rule matched and the default action was taken, this value is set to 0. If an error occurs during rules evaluation, it is set to -1. For any other error, it is set to -.
request_creation_time (22)	The time when the load balancer received the request from the client, in ISO 8601 format.
"actions_executed" (23)	The actions taken when processing the request, enclosed in double quotes. This value is a comma-separated list that can include the values described in Actions taken. If no action was taken, such as for a malformed request, this value is set to -.
"redirect_url" (24)	The URL of the redirect target for the location header of the HTTP response, enclosed in double quotes. If no redirect actions were taken, this value is set to -.
"error_reason" (25)	The error reason code, enclosed in double quotes. If the request failed, this is one of the error codes described in Error reason codes. If the actions taken do not include an authenticate action or the target is not a Lambda function, this value is set to -.

Field (position)	Description
"target:port_list" (26)	A space-delimited list of IP addresses and ports for the targets that processed this request, enclosed in double quotes. Currently, this list can contain one item and it matches the target:port field. If the client didn't send a full request, the load balancer can't dispatch the request to a target, and this value is set to -. If the target is a Lambda function, this value is set to -. If the request is blocked by AWS WAF, this value is set to -.
"target_status_code_list" (27)	A space-delimited list of status codes from the responses of the targets, enclosed in double quotes. Currently, this list can contain one item and it matches the target_status_code field. This value is recorded only if a connection was established to the target and the target sent a response. Otherwise, it is set to -.
"classification" (28)	The classification for desync mitigation, enclosed in double quotes. If the request does not comply with RFC 7230, the possible values are Acceptable, Ambiguous, and Severe. If the request complies with RFC 7230, this value is set to -.
"classification_reason" (29)	The classification reason code, enclosed in double quotes. If the request does not comply with RFC 7230, this is one of the classification codes described in Classification reasons. If the request complies with RFC 7230, this value is set to -.
conn_trace_id (30)	The connection traceability ID is a **unique opaque ID** used to identify each connection. After a connection is established with a client, subsequent requests from this client will contain this ID in their respective access log entries. This ID acts as a foreign key to create a link between the connection and access logs.

Field (position)	Description
"transfor med_host" (31)	The host header after it is modified by a host header rewrite transform. If any of the following are true, this value is set to -. • No transform was applied • The transform failed • The transform succeeded by there was no change to the host header • There is no original host header (for example, HTTP/1.0 requests)
"transfor med_uri" (32)	The URI after it is modified by a URL rewrite transform. If any of the following are true, this value is set to -. • No transform was applied • The transform failed • The transform succeeded by there was no change to the URI
"request_transform _status" (33)	The status of the rewrite transform. If no rewrite transform was applied, this value is set to -. Otherwise, this value is one of the status values described in the section called “Transform status codes”.

Actions taken

The load balancer stores the actions that it takes in the actions_executed field of the access log.

- `authenticate` — The load balancer validated the session, authenticated the user, and added the user information to the request headers, as specified by the rule configuration.
- `fixed-response` — The load balancer issued a fixed response, as specified by the rule configuration.
- `forward` — The load balancer forwarded the request to a target, as specified by the rule configuration.
- `redirect` — The load balancer redirected the request to another URL, as specified by the rule configuration.
- `rewrite` — The load balancer rewrote the request URL, as specified by the rule configuration.
- `waf` — The load balancer forwarded the request to AWS WAF to determine whether the request should be forwarded to the target. If this is the final action, AWS WAF determined that the

request should be rejected. By default, requests rejected by AWS WAF will be logged as "403" in the `elb_status_code` field. When AWS WAF is configured to reject requests with a Custom Response Code, the `elb_status_code` field will reflect the configured response code.

- `waf-failed` — The load balancer attempted to forward the request to AWS WAF, but this process failed.

Classification reasons

If a request does not comply with RFC 7230, the load balancer stores one of the following codes in the classification_reason field of the access log. For more information, see Desync mitigation mode.

Code	Description	Classification
`AmbiguousUri`	The request URI contains control characters.	Ambiguous
`BadContentLength`	The Content-Length header contains a value that cannot be parsed or is not a valid number.	Severe
`BadHeader`	A header contains a null character or carriage return.	Severe
`BadTransferEncoding`	The Transfer-Encoding header contains a bad value.	Severe
`BadUri`	The request URI contains a null character or carriage return.	Severe
`BadMethod`	The request method is malformed.	Severe
`BadVersion`	The request version is malformed.	Severe
`BothTeClPresent`	The request contains both a Transfer-Encoding header and a Content-Length header.	Ambiguous
`DuplicateContentLength`	There are multiple Content-Length headers with the same value.	Ambiguous
`EmptyHeader`	A header is empty or there is a line with only spaces.	Ambiguous

Code	Description	Classification
`GetHeadZeroContentLength`	There is a Content-Length header with a value of 0 for a GET or HEAD request.	Acceptable
`MultipleContentLength`	There are multiple Content-Length headers with different values.	Severe
`MultipleTransferEncodingChunked`	There are multiple Transfer-Encoding: chunked headers.	Severe
`NonCompliantHeader`	A header contains a non-ASCII or control character.	Acceptable
`NonCompliantVersion`	The request version contains a bad value.	Acceptable
`SpaceInUri`	The request URI contains a space that is not URL encoded.	Acceptable
`SuspiciousHeader`	There is a header that can be normalized to Transfer-Encoding or Content-Length using common text normalization techniques.	Ambiguous
`SuspiciousTeClPresent`	The request contains both a Transfer-Encoding header and a Content-Length header, with at least one of them being suspicious.	Severe
`UndefinedContentLengthSemantics`	There is a Content-Length header defined for a GET or HEAD request.	Ambiguous
`UndefinedTransferEncodingSemantics`	There is a Transfer-Encoding header defined for a GET or HEAD request.	Ambiguous

Error reason codes

If the load balancer cannot complete an authenticate action, the load balancer stores one of the following reason codes in the error_reason field of the access log. The load balancer also increments the corresponding CloudWatch metric. For more information, see Authenticate users using an Application Load Balancer.

Code	Description	Metric
`AuthInvalidCookie`	The authentication cookie is not valid.	`ELBAuthFailure`
`AuthInvalidGrantError`	The authorization grant code from the token endpoint is not valid.	`ELBAuthFailure`
`AuthInvalidIdToken`	The ID token is not valid.	`ELBAuthFailure`
`AuthInvalidStateParam`	The state parameter is not valid.	`ELBAuthFailure`
`AuthInvalidTokenResponse`	The response from the token endpoint is not valid.	`ELBAuthFailure`
`AuthInvalidUserinfoResponse`	The response from the user info endpoint is not valid.	`ELBAuthFailure`
`AuthMissingCodeParam`	The authentication response from the authorization endpoint is missing a query parameter named 'code'.	`ELBAuthFailure`
`AuthMissingHostHeader`	The authentication response from the authorization endpoint is missing a host header field.	`ELBAuthError`

Code	Description	Metric
AuthMissingStateParam	The authentication response from the authorization endpoint is missing a query parameter named 'state'.	ELBAuthFailure
AuthTokenEpRequestFailed	There is an error response (non-2XX) from the token endpoint.	ELBAuthError
AuthTokenEpRequestTimeout	The load balancer is unable to communicate with the token endpoint, or the token endpoint is not responding within 5 seconds.	ELBAuthError
AuthUnhandledException	The load balancer encountered an unhandled exception.	ELBAuthError
AuthUserinfoEpRequestFailed	There is an error response (non-2XX) from the IdP user info endpoint.	ELBAuthError
AuthUserinfoEpRequestTimeout	The load balancer is unable to communicate with the IdP user info endpoint, or the user info endpoint is not responding within 5 seconds.	ELBAuthError
AuthUserinfoResponseSizeExceeded	The size of the claims returned by the IdP exceeded 11K bytes.	ELBAuthUserClaimsSizeExceeded

If the load balancer cannot complete an jwt-validation action, the load balancer stores one of the following reason codes in the error_reason field of the access log. The load balancer also increments the corresponding CloudWatch metric. For more information, see Verify JWTs using an Application Load Balancer.

Code	Description	Metric
`JWTHeaderNotPresent`	Request does not contain Authorization header.	`JWTValidationFailureCount`
`JWTRequestFormatInvalid`	Token in request is malformed or missing mandatory parts (header, payload, or signature), Header does not contain "Bearer " prefix, Header contains a different auth type like "Basic ", Authorization header is present but token is not present, if there are multiple tokens present in the request	`JWTValidationFailureCount`
`AuthInvJWKSRequestTimeoutalidIdToken`	The load balancer is unable to communicate with the JWKS endpoint, or the JWKS endpoint is not responding within 5 seconds.	`JWTValidationFailureCount`
`JWKSResponseSizeExceeded`	The size of the response returned by the JWKS endpoint exceeds 150KB or the number of keys returned by the JWKS endpoint exceeds 10.	`JWTValidationFailureCount`
`AuthInvJWKSRequestFailed`	There is an error response (non-2XX) from the JWKS endpoint.	`JWTValidationFailureCount`
`AuthInvalidUseJWTSignatureValidationFailed`	Failed to validate token signature for any reason including signature does not match, the public key was invalid and could not be converted to a decoding key, public key size was not 2K, Token is signed with an Unsupported Algorithm, the KID in the token is not present in the JWKS endpoint.	`JWTValidationFailureCount`

Code	Description	Metric
JWTClaimNotPresent	JWT in the client request does not contain a claim which is required for validation	JWTValidationFailureCount
JWTClaimFormatInvalid	The format of the claim's value in the JWT does not match the format specified in the configuration	JWTValidationFailureCount
JWTClaimValueInvalid	The value of the claim in the JWT is invalid.	JWTValidationFailureCount
JWTValidationInternalError	The load balancer encountered an unexpected error while validating the JWT in the client request.	JWTValidationFailureCount

If a request to a weighted target group fails, the load balancer stores one of the following error codes in the error_reason field of the access log.

Code	Description
AWSALBTGCookieInvalid	The AWSALBTG cookie, which is used with weighted target groups, is not valid. For example, the load balancer returns this error when cookie values are URL encoded.
WeightedTargetGroupsUnhandledException	The load balancer encountered an unhandled exception.

If a request to a Lambda function fails, the load balancer stores one of the following reason codes in the error_reason field of the access log. The load balancer also increments the corresponding CloudWatch metric. For more information, see the Lambda Invoke action.

Code	Description	Metric
LambdaAccessDenied	The load balancer did not have permission to invoke the Lambda function.	LambdaUserError
LambdaBadRequest	Lambda invocation failed because the client request headers or body did not contain only UTF-8 characters.	LambdaUserError
LambdaConnectionError	The load balancer cannot connect to Lambda.	LambdaInternalError
LambdaConnectionTimeout	An attempt to connect to Lambda timed out.	LambdaInternalError
LambdaEC2AccessDeniedException	Amazon EC2 denied access to Lambda during function initialization.	LambdaUserError
LambdaEC2ThrottledException	Amazon EC2 throttled Lambda during function initialization.	LambdaUserError
LambdaEC2UnexpectedException	Amazon EC2 encountered an unexpected exception during function initialization.	LambdaUserError
LambdaENILimitReachedException	Lambda couldn't create a network interface in the VPC specified in the configuration of the Lambda function because the limit for network interfaces was exceeded.	LambdaUserError
LambdaInvalidResponse	The response from the Lambda function is malformed or is missing required fields.	LambdaUserError
LambdaInvalidRuntimeException	The specified version of the Lambda runtime is not supported.	LambdaUserError

Code	Description	Metric
LambdaInvalidSecurityGroupIDException	The security group ID specified in the configuration of the Lambda function is not valid.	LambdaUserError
LambdaInvalidSubnetIDException	The subnet ID specified in the configuration of the Lambda function is not valid.	LambdaUserError
LambdaInvalidZipFileException	Lambda could not unzip the specified function zip file.	LambdaUserError
LambdaKMSAccessDeniedException	Lambda could not decrypt environment variables because access to the KMS key was denied. Check the KMS permissions of the Lambda function.	LambdaUserError
LambdaKMSDisabledException	Lambda could not decrypt environment variables because the specified KMS key is disabled. Check the KMS key settings of the Lambda function.	LambdaUserError
LambdaKMSInvalidStateException	Lambda could not decrypt environment variables because the state of the KMS key is not valid. Check the KMS key settings of the Lambda function.	LambdaUserError
LambdaKMSNotFoundException	Lambda could not decrypt environment variables because the KMS key was not found. Check the KMS key settings of the Lambda function.	LambdaUserError
LambdaRequestTooLarge	The size of the request body exceeded 1 MB.	LambdaUserError

Code	Description	Metric
LambdaResourceNotFound	The Lambda function could not be found.	LambdaUserError
LambdaResponseTooLarge	The size of the response exceeded 1 MB.	LambdaUserError
LambdaServiceException	Lambda encountered an internal error.	LambdaInternalError
LambdaSubnetIPAddressLimitReachedException	Lambda could not set up VPC access for the Lambda function because one or more subnets have no available IP addresses.	LambdaUserError
LambdaThrottling	The Lambda function was throttled because there were too many requests.	LambdaUserError
LambdaUnhandled	The Lambda function encountered an unhandled exception.	LambdaUserError
LambdaUnhandledException	The load balancer encountered an unhandled exception.	LambdaInternalError
LambdaWebsocketNotSupported	WebSockets are not supported with Lambda.	LambdaUserError

If the load balancer encounters an error when forwarding requests to AWS WAF, it stores one of the following error codes in the error_reason field of the access log.

Code	Description
WAFConnectionError	The load balancer cannot connect to AWS WAF.

Code	Description
WAFConnectionTimeout	The connection to AWS WAF timed out.
WAFResponseReadTimeout	A request to AWS WAF timed out.
WAFServiceError	AWS WAF returned a 5XX error.
WAFUnhandledException	The load balancer encountered an unhandled exception.

Transform status codes

Code	Description
TransformBufferTooSmall	The rewrite transform failed because the result exceeded the size of an internal buffer. Try to make the regular expression less complex.
TransformCompileError	The compilation of the regular expression failed.
TransformCompileTooBig	The compiled regular expression was too large. Try to make the regular expression less complex.
TransformInvalidHost	The host header rewrite transform failed because the resulting host is not valid.
TransformInvalidPath	The URL rewrite transform failed because the resulting path is not valid.
TransformRegexSyntaxError	The regular expression contained a syntax error.
TransformReplaceError	The transform replacement failed.

Code	Description
`TransformSuccess`	The rewrite transform completed successfully.

Example log entries

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

Example HTTP Entry

The following is an example log entry for an HTTP listener (port 80 to port 80):

```
http 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 10.0.0.1:80 0.000 0.001 0.000 200 200 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337262-36d228ad5d99923122bbe354" "-" "-"
0 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.0.1:80" "200" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

Example HTTPS Entry

The following is an example log entry for an HTTPS listener (port 443 to port 80):

```
https 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 10.0.0.1:80 0.086 0.048 0.037 200 200 0 57
"GET https://www.example.com:443/ HTTP/1.1" "curl/7.46.0" ECDHE-RSA-AES128-GCM-SHA256
 TLSv1.2
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337281-1d84f3d73c47ec4e58577259" "www.example.com" "arn:aws:acm:us-
east-2:123456789012:certificate/12345678-1234-1234-1234-123456789012"
1 2018-07-02T22:22:48.364000Z "authenticate,forward" "-" "-" "10.0.0.1:80" "200" "-"
 "-"
TID_1234abcd5678ef90 "m.example.com" "-" "TransformSuccess"
```

Example HTTP/2 Entry

The following is an example log entry for an HTTP/2 stream.

```
h2 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
10.0.1.252:48160 10.0.0.66:9000 0.000 0.002 0.000 200 200 5 257
"GET https://10.0.2.105:773/ HTTP/2.0" "curl/7.46.0" ECDHE-RSA-AES128-GCM-SHA256
 TLSv1.2
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337327-72bd00b0343d75b906739c42" "-" "-"
1 2018-07-02T22:22:48.364000Z "redirect" "https://example.com:80/" "-" "10.0.0.66:9000"
 "200" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

Example WebSockets Entry

The following is an example log entry for a WebSockets connection.

```
ws 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
10.0.0.140:40914 10.0.1.192:8010 0.001 0.003 0.000 101 101 218 587
"GET http://10.0.0.30:80/ HTTP/1.1" "-" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
1 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.1.192:8010" "101" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

Example Secured WebSockets Entry

The following is an example log entry for a secured WebSockets connection.

```
wss 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
10.0.0.140:44244 10.0.0.171:8010 0.000 0.001 0.000 101 101 218 786
"GET https://10.0.0.30:443/ HTTP/1.1" "-" ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2
arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
1 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.0.171:8010" "101" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

Example Entries for Lambda Functions

The following is an example log entry for a request to a Lambda function that succeeded:

```
http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
```

```
192.168.131.39:2817 - 0.000 0.001 0.000 200 200 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
0 2018-11-30T22:22:48.364000Z "forward" "-" "-" "-" "-" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

The following is an example log entry for a request to a Lambda function that failed:

```
http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 - 0.000 0.001 0.000 502 - 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
0 2018-11-30T22:22:48.364000Z "forward" "-" "LambdaInvalidResponse" "-" "-" "-" "-"
TID_1234abcd5678ef90 "-" "-" "-"
```

Configure log delivery notifications

To receive notifications when ELB delivers logs to your S3 bucket, use Amazon S3 Event Notifications. ELB uses PutObject, CreateMultipartUpload, and POST Object to deliver logs to Amazon S3. To ensure that you receive all log delivery notifications, include all of these object creation events in your configuration.

For more information, see Amazon S3 Event Notifications in the *Amazon Simple Storage Service User Guide*.

Processing access log files

The access log files are compressed. 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 Application Load Balancer logs in the *Amazon Athena User Guide*.
- Loggly
- Splunk
- Sumo logic

Enable access logs for your Application Load Balancer

When you enable access logs 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 ELB permission to write to the bucket.

Tasks

- Step 1: Create an S3 bucket
- Step 2: Attach a policy to your S3 bucket
- Step 3: Configure access logs
- Step 4: Verify bucket permissions
- Troubleshooting

Step 1: Create an S3 bucket

When you enable access logs, you must specify an S3 bucket for the access logs. 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 only server-side encryption option that's supported is Amazon S3-managed keys (SSE-S3). For more information, see Amazon S3-managed encryption keys (SSE-S3).

To create an S3 bucket using the Amazon S3 console

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.

2. Choose **Create bucket**.
3. On the **Create bucket** page, do the following:

   a. For **Bucket name**, enter a name for your bucket. This name must be unique across all existing bucket names in Amazon S3. In some Regions, there might be additional restrictions on bucket names. For more information, see Bucket restrictions and limitations in the *Amazon S3 User Guide*.

   b. For **AWS Region**, select the Region where you created your load balancer.

   c. For **Default encryption**, choose **Amazon S3-managed keys (SSE-S3)**.

   d. Choose **Create bucket**.

Step 2: Attach a policy to your S3 bucket

Your S3 bucket must have a bucket policy that grants ELB permission to write the access logs to the bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. Each statement includes information about a single permission and contains a series of elements.

If you're using an existing bucket that already has an attached policy, you can add the statement for ELB access logs to the policy. If you do so, we recommend that you evaluate the resulting set of permissions to ensure that they are appropriate for the users that need access to the bucket for access logs.

Bucket policy

This policy grants permissions to the log delivery service.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
```

```
}
```

For `Resource`, enter the ARN of the location for the access logs, using the format shown in the example policy. Always include the account ID of the account with the load balancer in the resource path of the S3 bucket ARN. This ensures that only load balancers from the specified account can write access logs to the S3 bucket.

The ARN that you specify depends on whether you plan to include a prefix when you enable access logs in step 3.

Example S3 bucket ARN with a prefix

The S3 bucket name is amzn-s3-demo-logging-bucket and the prefix is logging-prefix.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

Example S3 bucket ARN with no prefix

The S3 bucket name is amzn-s3-demo-logging-bucket. There is no prefix portion in the S3 bucket ARN.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

Security best practices

- Use the full resource path, including the account ID portion of the S3 bucket ARN. Don't use wildcards (*) in the account ID portion of the S3 bucket ARN.

  ```
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
  ```

- Use `aws:SourceArn` to ensure that only load balancers from the specified Region and account can use your bucket.

  ```
  "Condition": {
      "ArnLike": {
          "aws:SourceArn":
   "arn:aws:elasticloadbalancing:region:123456789012:loadbalancer/*"
      }
  }
  ```

- Use `aws:SourceOrgId` with `aws:SourceArn` to ensure that only load balancers from the specified organization can use your bucket.

```
"Condition": {
    "StringEquals": {
        "aws:SourceOrgId": "o-1234567890"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:elasticloadbalancing:*:*:loadbalancer/*"
    }
}
```

- If you have a `Deny` statement to prevent access to service principals except those explicitly allowed, be sure to add `logdelivery.elasticloadbalancing.amazonaws.com` to the list of allowed service principals. For example, if you used the `aws:PrincipalServiceNamesList` condition, add `logdelivery.elasticloadbalancing.amazonaws.com` as follows:

```
{
  "Effect": "Deny",
  "Principal": "*",
  "Condition": {
      "StringNotEqualsIfExists": {
          "aws:PrincipalServiceNamesList": [
              "logdelivery.elasticloadbalancing.amazonaws.com",
              "service.amazonaws.com"
          ]
      }
  }
}
```

  If you used the `NotPrincipal` element, add `logdelivery.elasticloadbalancing.amazonaws.com` as follows.
  Note that we recommend that you use the `aws:PrincipalServiceName` or `aws:PrincipalServiceNamesList` condition key to explicitly allow service principals instead of using the `NotPrincipal` element. For more information, see NotPrincipal.

```
{
  "Effect": "Deny",
  "NotPrincipal": {
    "Service": [
       "logdelivery.elasticloadbalancing.amazonaws.com",
```

```
        "service.amazonaws.com"
    ]
  }
},
```

After you create your bucket policy, use an Amazon S3 interface, such as the Amazon S3 console or AWS CLI commands, to attach your bucket policy to your S3 bucket.

Console

To attach your bucket policy to your S3 bucket

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket to open its details page.
3. Choose **Permissions** and then choose **Bucket policy**, **Edit**.
4. Update the bucket policy to grant the required permissions.
5. Choose **Save changes**.

AWS CLI

To attach your bucket policy to your S3 bucket

Use the put-bucket-policy command. In this example, the bucket policy was saved to the specified .json file.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://access-log-policy.json
```

Step 3: Configure access logs

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

Requirements

The bucket must meet the requirements described in step 1, and you must attach a bucket policy as described in step 2. If you include a prefix, it must not include the string "AWSLogs".

To manage the S3 bucket for your access logs

Be sure to disable access logs before you delete the bucket that you configured for access logs. Otherwise, if there is a new bucket with the same name and the required bucket policy but created in an AWS account that you don't own, ELB could write the access logs for your load balancer to this new bucket.

Console

To enable access logs

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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-alb
      Type: application
      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"
```

Step 4: Verify bucket permissions

After access logs are enabled for your load balancer, ELB validates the S3 bucket and creates a test file to ensure that the bucket policy specifies the required permissions. You can use the Amazon S3 console to verify that the test file was created. The test file is not an actual access log file; it doesn't contain example records.

To verify a test file was created in your bucket using the Amazon S3 console

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket that you specified for access logs.
3. Navigate to the test file, `ELBAccessLogTestFile`. The location depends on whether you're using a prefix.
   - Location with a prefix: *amzn-s3-demo-logging-bucket*/*logging-prefix*/AWSLogs/*123456789012*/ELBAccessLogTestFile

- Location without a prefix: *amzn-s3-demo-logging-bucket*/AWSLogs/*123456789012*/ELBAccessLogTestFile

Troubleshooting

If you receive an access denied error, the following are possible causes:

- The bucket policy does not grant ELB permission to write access logs to the bucket. Verify that you are using the correct bucket policy for the Region. Verify that the resource ARN uses the same bucket name that you specified when you enabled access logs. Verify that the resource ARN does not include a prefix if you did not specify a prefix when you enabled access logs.
- The bucket uses an unsupported server-side encryption option. The bucket must use Amazon S3-managed keys (SSE-S3).

Disable access logs for your Application Load Balancer

You can disable access logs for your load balancer at any time. After you disable access logs, your access logs remain in your S3 bucket until you delete 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://eusc-de-east-1.console.amazonaws-eusc.eu/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
```

Connection logs for your Application Load Balancer

ELB provides connection logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the client's IP address and port, listener port, the TLS cipher and protocol used, TLS handshake latency, connection status, and client certificate details. You can use these connection logs to analyze request patterns and troubleshoot issues.

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

You are charged storage costs for Amazon S3, but not charged for the bandwidth used by ELB to send log files to Amazon S3. For more information about storage costs, see Amazon S3 pricing.

Contents

Connection log files

ELB 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 connection logs use the following format:

```
bucket[/prefix]/AWSLogs/aws-account-id/elasticloadbalancing/region/yyyy/mm/dd/
conn_log_aws-account-id_elasticloadbalancing_region_app.load-balancer-id_end-time_ip-
address_random-string.log.gz
```

bucket

The name of the S3 bucket.

prefix

(Optional) The prefix (logical hierarchy) for the bucket. The prefix that you specify must not include the string `AWSLogs`. For more information, see Organizing objects using prefixes.

`AWSLogs`

We add the portion of the file name starting with `AWSLogs` after the bucket name and optional prefix that you specify.

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 20140215T2340Z contains entries for requests made between 23:35 and 23:40 in UTC or Zulu time.

ip-address

The IP address of the load balancer node that handled the request. For an internal load balancer, this is a private IP address.

random-string

A system-generated random string.

The following is an example log file name with a prefix:

```
s3://amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/
elasticloadbalancing/us-east-2/2022/05/01/
conn_log_123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

The following is an example log file name without a prefix:

```
s3://amzn-s3-demo-logging-bucket/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/conn_log_123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_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 Object lifecycle management in the *Amazon S3 User Guide*.

Connection log entries

Each connection attempt has an entry in a connection log file. How client requests are sent is determined by the connection being persistent, or nonpersistent. Nonpersistent connections have a single request, which creates a single entry in the access log and connection log. Persistent connections have multiple requests, which creates multiple entries in the access log and a single entry in the connection log.

Contents

Syntax

The following table describes the fields of a connection log entry, in order. All fields are delimited by spaces. When we add a new field, we add it to the end of the log entry. As we prepare to release a new field, you might see an additional trailing "-" before the field is released. Ensure that you

configure log parsing to stop after the last documented field, and update log parsing after we release a new field.

Field (position)	Description
timestamp (1)	The time, in ISO 8601 format, when the load balancer successfully established or failed to establish a connection.
client_ip (2)	The IP address of the requesting client.
client_port (3)	The port of the requesting client.
listener_port (4)	The port of the load balancer listener receiving the client request.
tls_protocol (5)	[HTTPS listener] The SSL/TLS protocol used during handshakes. This field is set to `-` for non SSL/TLS requests.
tls_cipher (6)	[HTTPS listener] The SSL/TLS protocol used during handshakes. This field is set to `-` for non SSL/TLS requests.
tls_handshake_latency (7)	[HTTPS listener] The total time in seconds, with a millisecond precision, elapsed while establishing a successful handshake. This field is set to `-` when: • The incoming request is not a SSL/TLS request. • The handshake is not established successfully.
leaf_client_cert_subject (8)	[HTTPS listener] The subject name of the leaf client certificate. This field is set to `-` when: • The incoming request is not a SSL/TLS request. • The load balancer listener is not configured with mTLS enabled. • The server is not able to load/parse the leaf client certificate.
leaf_client_cert_validity (9)	[HTTPS listener] The validity, with `not-before` and `not-after` in ISO 8601 format, of the leaf client certificate. This field is set to `-` when: • The incoming request is not a SSL/TLS request.

Field (position)	Description
	• The load balancer listener is not configured with mTLS enabled. • The server is not able to load/parse the leaf client certificate.
leaf_client_cert_serial_number (10)	[HTTPS listener] The serial number of the leaf client certificate. This field is set to `-` when: • The incoming request is not a SSL/TLS request. • The load balancer listener is not configured with mTLS enabled. • The server is not able to load/parse the leaf client certificate.
tls_verify_status (11)	[HTTPS listener] The status of the connection request. This value is `Success` if the connection is established successfully. On an unsuccessful connection the value is `Failed:$error_code` .
conn_trace_id (12)	The connection traceability ID is a **unique opaque ID** used to identify each connection. After a connection is established with a client, subsequent requests from this client contain this ID in their respective access log entries. This ID acts as a foreign key to create a link between the connection and access logs.
tls_keyexchange (13)	[HTTPS listener] The key exchange used during handshakes for TLS or PQ-TLS . This field is set to `-` for non SSL/TLS requests.

Error reason codes

If the load balancer is unable to establish a connection, the load balancer stores one of the following reason codes in the connection log.

Code	Description
`ClientCertMaxChainDepthExceeded`	The maximum client certificate chain depth has been exceeded

Code	Description	
ClientCertMaxSizeExceeded	The maximum client certificate size has been exceeded	
ClientCertCrlHit	Client certificate has been revoked by the CA	
ClientCertCrlProcessingError	CRL processing error	
ClientCertUntrusted	Client certificate is untrusted	
ClientCertNotYetValid	Client certificate is not yet valid	
ClientCertExpired	Client certificate is expired	
ClientCertTypeUnsupported	Client certificate type is unsupported	
ClientCertInvalid	Client certificate is invalid	
ClientCertPurposeInvalid	Client certificate purpose is invalid	
ClientCertRejected	Client certificate is rejected by custom server validation	
UnmappedConnectionError	Unmapped runtime connection error	

Example log entries

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

The following is an example log entry for a successful connection with a HTTPS listener with mutual TLS verify mode enabled on port 443.

```
2023-10-04T17:05:15.514108Z 203.0.113.1 36280 443 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
 4.036
"CN=amazondomains.com,O=endEntity,L=Seattle,ST=Washington,C=US"
 NotBefore=2023-09-21T22:43:21Z;NotAfter=2026-06-17T22:43:21Z
FEF257372D5C14D4 Success TID_3180a73013c8ca4bac2f731159d4b0fe
```

The following is an example log entry for a failed connection with a HTTPS listener with mutual TLS verify mode enabled on port 443.

```
2023-10-04T17:05:15.514108Z 203.0.113.1 36280 443 TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256
 -
"CN=amazondomains.com,O=endEntity,L=Seattle,ST=Washington,C=US"
 NotBefore=2023-09-21T22:43:21Z;NotAfter=2026-06-17T22:43:21Z
FEF257372D5C14D4 Failed:ClientCertUntrusted TID_1c71a68d70587445ad5127ff8b2687d7
```

Processing connection log files

The connection 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 connection logs:

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.
- Loggly
- Splunk

- Sumo logic

Enable connection logs for your Application Load Balancer

When you enable connection logs 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 ELB permission to write to the bucket.

Tasks

- Step 1: Create an S3 bucket
- Step 2: Attach a policy to your S3 bucket
- Step 3: Configure connection logs
- Step 4: Verify bucket permissions
- Troubleshooting

Step 1: Create an S3 bucket

When you enable connection logs, you must specify an S3 bucket for the connection logs. You can use an existing bucket, or create a bucket specifically for connection 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 only server-side encryption option that's supported is Amazon S3-managed keys (SSE-S3). For more information, see Amazon S3-managed encryption keys (SSE-S3).

To create an S3 bucket using the Amazon S3 console

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Choose **Create bucket**.
3. On the **Create bucket** page, do the following:

    a. For **Bucket name**, enter a name for your bucket. This name must be unique across all existing bucket names in Amazon S3. In some Regions, there might be additional

restrictions on bucket names. For more information, see Bucket restrictions and limitations in the *Amazon S3 User Guide*.

b. For **AWS Region**, select the Region where you created your load balancer.

c. For **Default encryption**, choose **Amazon S3-managed keys (SSE-S3)**.

d. Choose **Create bucket**.

Step 2: Attach a policy to your S3 bucket

Your S3 bucket must have a bucket policy that grants ELB permission to write the connection logs to the bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. Each statement includes information about a single permission and contains a series of elements.

If you're using an existing bucket that already has an attached policy, you can add the statement for ELB connection logs to the policy. If you do so, we recommend that you evaluate the resulting set of permissions to ensure that they are appropriate for the users that need access to the bucket for connection logs.

Bucket policy

This policy grants permissions to the specified log delivery service.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
}
```

For `Resource`, enter the ARN of the location for the access logs, using the format shown in the example policy. Always include the account ID of the account with the load balancer in the resource

path of the S3 bucket ARN. This ensures that only load balancers from the specified account can write access logs to the S3 bucket.

The ARN that you specify depends on whether you plan to include a prefix when you enable access logs in step 3.

Example S3 bucket ARN with a prefix

The S3 bucket name is amzn-s3-demo-logging-bucket and the prefix is logging-prefix.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

Example S3 bucket ARN with no prefix

The S3 bucket name is amzn-s3-demo-logging-bucket. There is no prefix portion in the S3 bucket ARN.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

Security best practices

To enhance security, use precise S3 bucket ARNs.

- Use the full resource path, not just the S3 bucket ARN.
- Include the account ID portion of the S3 bucket ARN.
- Don't use wildcards (*) in the account ID portion of the S3 bucket ARN.

After you create your bucket policy, use an Amazon S3 interface, such as the Amazon S3 console or AWS CLI commands, to attach your bucket policy to your S3 bucket.

Console

To attach your bucket policy to your S3 bucket

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket to open its details page.
3. Choose **Permissions** and then choose **Bucket policy**, **Edit**.
4. Update the bucket policy to grant the required permissions.

5. Choose **Save changes**.

AWS CLI

To attach your bucket policy to your S3 bucket

Use the put-bucket-policy command. In this example, the bucket policy was saved to the specified .json file.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://access-log-policy.json
```

Step 3: Configure connection logs

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

Requirements

The bucket must meet the requirements described in step 1, and you must attach a bucket policy as described in step 2. If you specify a prefix, it must not include the string "AWSLogs".

To manage the S3 bucket for your connection logs

Be sure to disable connection logs before you delete the bucket that you configured for connection logs. Otherwise, if there is a new bucket with the same name and the required bucket policy but created in an AWS account that you don't own, ELB could write the connection logs for your load balancer to this new bucket.

Console

To enable connection logs

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 **Connection 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://bucket-name/prefix`
   - URI without a prefix: `s3://bucket-name`
7. Choose **Save changes**.

AWS CLI

To enable connection 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=connection_logs.s3.enabled,Value=true \
        Key=connection_logs.s3.bucket,Value=amzn-s3-demo-logging-bucket \
        Key=connection_logs.s3.prefix,Value=logging-prefix
```

CloudFormation

To enable connection logs

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

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

```
    LoadBalancerAttributes:
      - Key: "connection_logs.s3.enabled"
        Value: "true"
      - Key: "connection_logs.s3.bucket"
        Value: "amzn-s3-demo-logging-bucket"
      - Key: "connection_logs.s3.prefix"
        Value: "logging-prefix"
```

Step 4: Verify bucket permissions

After connection logs are enabled for your load balancer, ELB validates the S3 bucket and creates a test file to ensure that the bucket policy specifies the required permissions. You can use the Amazon S3 console to verify that the test file was created. The test file is not an actual connection log file; it doesn't contain example records.

To verify that ELB created a test file in your S3 bucket

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket that you specified for connection logs.
3. Navigate to the test file, `ELBConnectionLogTestFile`. The location depends on whether you're using a prefix.
   - Location with a prefix: *amzn-s3-demo-logging-bucket*/*prefix*/AWSLogs/*123456789012*/ELBConnectionLogTestFile
   - Location without a prefix: *amzn-s3-demo-logging-bucket*/AWSLogs/*123456789012*/ELBConnectionLogTestFile

Troubleshooting

If you receive an access denied error, the following are possible causes:

- The bucket policy does not grant ELB permission to write connection logs to the bucket. Verify that you are using the correct bucket policy for the Region. Verify that the resource ARN uses the same bucket name that you specified when you enabled connection logs. Verify that the resource ARN does not include a prefix if you did not specify a prefix when you enabled connection logs.
- The bucket uses an unsupported server-side encryption option. The bucket must use Amazon S3-managed keys (SSE-S3).

Disable connection logs for your Application Load Balancer

You can disable connection logs for your load balancer at any time. After you disable connection logs, your connection logs remain in your S3 bucket until you delete them. For more information, see Creating, configuring, and working with buckets in the *Amazon S3 User Guide*.

Console

To disable connection logs

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 **Connection logs**.
6. Choose **Save changes**.

AWS CLI

To disable connection logs

Use the modify-load-balancer-attributes command.

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

Health check logs

ELB provides health check logs that capture detailed information about the health check status of your registered targets, including failure reasons when health checks fail. Health check logs are supported for EC2 instances, IP address, and Lambda function targets. Each log entry contains information such as the health check request type or connection, timestamp, target address, target group ID, health status and reason code. You can use these health check logs to analyze target health patterns, monitor health transitions, and troubleshoot issues.

Health check logs are an optional feature that is disabled by default. After you enable health check logs for your load balancer, ELB captures the logs and stores them as compressed files in the Amazon S3 bucket that you specify. You can disable health check logs at any time.

You are charged storage costs for Amazon S3, but not charged for the bandwidth used by ELB to send log files to Amazon S3. For more information about storage costs, see Amazon S3 pricing.

Contents

Health check log files

ELB publishes a log file for each load balancer node every 5 minutes. The load balancer can deliver multiple logs for the same period when a large number of targets are attached to the load balancer or a small health check interval is configured (for example, every 5 seconds).

The file names of the health check logs use the following format:

```
bucket[/prefix]/AWSLogs/aws-account-id/elasticloadbalancing/region/yyyy/mm/dd/
health_check_log_aws-account-id_elasticloadbalancing_region_app.load-balancer-id_end-
time_ip-address_random-string.log.gz
```

bucket

The name of the S3 bucket.

prefix

(Optional) The prefix (logical hierarchy) for the bucket. The prefix that you specify must not include the string `AWSLogs`. For more information, see Organizing objects using prefixes.

`AWSLogs`

We add the portion of the file name starting with `AWSLogs` after the bucket name and optional prefix that you specify.

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 20140215T2340Z contains entries for requests made between 23:35 and 23:40 in UTC or Zulu time.

ip-address

The IP address of the load balancer node that handled the request. For an internal load balancer, this is a private IP address.

random-string

A system-generated random string.

The following is an example log file name with a prefix:

```
s3://amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/
elasticloadbalancing/us-east-2/2022/05/01/
health_check_log_123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

The following is an example log file name without a prefix:

```
s3://amzn-s3-demo-logging-bucket/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/health_check_log_123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_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 Object lifecycle management in the *Amazon S3 User Guide*.

Health check log entries

ELB logs target health check results including the failure reasons for all registered targets of that load balancer. Each log entry contains the details of a single health check result made to the registered target.

Contents

Syntax

The following table describes the fields of a health check log entry, in order. All fields are delimited by spaces. When we add a new field, we add it to the end of the log entry. As we prepare to release a new field, you might see an additional trailing "-" before the field is released. Ensure that you configure log parsing to stop after the last documented field, and update log parsing after we release a new field.

Field (position)	Description
type (1)	The type of health check request or connection. The possible values are as follows (ignore any other values): • `http` -- HTTP • `https` -- HTTP over TLS • h2 -- HTTP/2 over TLS • `grpc` -- gRPC • `lambda` -- Lambda Function

Field (position)	Description
time (2)	Timestamp of when health check is initiated on a target, in ISO 8601 format.
latency (3)	Total time elapsed#(in seconds) to complete the current health check.
target_addr (4)	IP address and port of the target in the format, IP:Port. Lambda's ARN if the target is a Lambda function.
target_group_id (5)	Name of the target group the target is associated with.
status (6)	The status of the health check. This value isPASS#if the health check succeeds. On an unsuccessful health check the value is`FAIL`
status_code (7)	The response code received from the target for the health check request.
reason_code (8)	The reason for failure if the health check fails. See Error reason codes

Error reason codes

If the target health check fails, the load balancer will log one of the following reason codes in the health check log.

Code	Description
`RequestTimedOut`	Health check request timed out while waiting for response
`ConnectionTimedOut`	Health check failed because TCP connection attempt timed out
`ConnectionReset`	Health check failed due to connection reset
`ResponseCodeMismatch`	HTTP status code of the target's response to the health check request did not match the configured status code

Code	Description	
ResponseStringMismatch	Response body returned by the target did not contain the string configured in the target group health check configuration	
InternalError	Internal load balancer error	
TargetError	Target returns 5xx error code in response to the health check request	
GRPCStatusHeaderEmpty	GRPC target response has a grpc-status header without value	
GRPCUnexpectedStatus	GRPC target responds with an unexpected grpc-status	

Example log entries

The following are examples of health check log entries. Note that the example text appears on multiple lines only to make them easier to read.

The following is an example log entry for a successful health check.

```
http 2025-10-31T12:44:59.875678Z 0.019584011 172.31.20.97:80 HCLogsTestIPs PASS 200 -
```

The following is an example log entry for a failed health check.

```
http 2025-10-31T12:44:58.901409Z 1.121980746 172.31.31.9:80 HCLogsTestIPs FAIL 502
 TargetError
```

Configure log delivery notifications

To receive notifications when ELB delivers logs to your S3 bucket, use Amazon S3 Event Notifications. ELB uses PutObject, CreateMultipartUpload, and POST Object to deliver logs to Amazon S3. To ensure that you receive all log delivery notifications, include all of these object creation events in your configuration.

For more information, see Amazon S3 Event Notifications in the *Amazon Simple Storage Service User Guide*.

Processing health check log files

The health check log files are compressed. 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 health-check logs:

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.
- Loggly
- Splunk
- Sumo logic

Enable health check logs for your Application Load Balancer

When you enable health check logs 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 ELB permission to write to the bucket.

Tasks

- Step 1: Create an S3 bucket
- Step 2: Attach a policy to your S3 bucket
- Step 3: Configure health check logs
- Step 4: Verify bucket permissions
- Troubleshooting

Step 1: Create an S3 bucket

When you enable health-check logs, you must specify an S3 bucket for the health-check logs. You can use an existing bucket, or create a bucket specifically for health-check 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 only server-side encryption option that's supported is Amazon S3-managed keys (SSE-S3). For more information, see Amazon S3-managed encryption keys (SSE-S3).

To create an S3 bucket using the Amazon S3 console

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Choose **Create bucket**.
3. On the **Create bucket** page, do the following:
   a. For **Bucket name**, enter a name for your bucket. This name must be unique across all existing bucket names in Amazon S3. In some Regions, there might be additional restrictions on bucket names. For more information, see Bucket restrictions and limitations in the *Amazon S3 User Guide*.
   b. For **AWS Region**, select the Region where you created your load balancer.
   c. For **Default encryption**, choose **Amazon S3-managed keys (SSE-S3)**.
   d. Choose **Create bucket**.

Step 2: Attach a policy to your S3 bucket

Your S3 bucket must have a bucket policy that grants ELB permission to write the health check logs to the bucket. Bucket policies are a collection of JSON statements written in the access policy language to define access permissions for your bucket. Each statement includes information about a single permission and contains a series of elements.

If you're using an existing bucket that already has an attached policy, you can add the statement for ELB health check logs to the policy. If you do so, we recommend that you evaluate the resulting

set of permissions to ensure that they are appropriate for the users that need access to the bucket for health check logs.

Bucket policy

This policy grants permissions to the specified log delivery service.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
}
```

For `Resource`, enter the ARN of the location for the access logs, using the format shown in the example policy. Always include the account ID of the account with the load balancer in the resource path of the S3 bucket ARN. This ensures that only load balancers from the specified account can write access logs to the S3 bucket.

The ARN that you specify depends on whether you plan to include a prefix when you enable access logs in step 3.

Example S3 bucket ARN with a prefix

The S3 bucket name is amzn-s3-demo-logging-bucket and the prefix is logging-prefix.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

Example S3 bucket ARN with no prefix

The S3 bucket name is amzn-s3-demo-logging-bucket. There is no prefix portion in the S3 bucket ARN.

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

Security best practices

To enhance security, use precise S3 bucket ARNs.

- Use the full resource path, not just the S3 bucket ARN.
- Include the account ID portion of the S3 bucket ARN.
- Don't use wildcards (*) in the account ID portion of the S3 bucket ARN.

After you create your bucket policy, use an Amazon S3 interface, such as the Amazon S3 console or AWS CLI commands, to attach your bucket policy to your S3 bucket.

Console

To attach your bucket policy to your S3 bucket

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket to open its details page.
3. Choose **Permissions** and then choose **Bucket policy**, **Edit**.
4. Update the bucket policy to grant the required permissions.
5. Choose **Save changes**.

AWS CLI

To attach your bucket policy to your S3 bucket

Use the put-bucket-policy command. In this example, the bucket policy was saved to the specified .json file.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://access-log-policy.json
```

Step 3: Configure health check logs

Use the following procedure to configure health check logs to capture and deliver log files to your S3 bucket.

Requirements

The bucket must meet the requirements described in step 1, and you must attach a bucket policy as described in step 2. If you specify a prefix, it must not include the string "AWSLogs".

To manage the S3 bucket for your health check logs

Be sure to disable health check logs before you delete the bucket that you configured for health check logs. Otherwise, if there is a new bucket with the same name and the required bucket policy but created in an AWS account that you don't own, ELB could write the health check logs for your load balancer to this new bucket.

Console

To enable health check logs

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 **Health Check 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://bucket-name/prefix`
   - URI without a prefix: `s3://bucket-name`
7. Choose **Save changes**.

AWS CLI

To enable health check 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=health_check_logs.s3.enabled,Value=true \
        Key=health_check_logs.s3.bucket,Value=amzn-s3-demo-logging-bucket \
        Key=health_check_logs.s3.prefix,Value=logging-prefix
```

CloudFormation

To enable health check logs

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

```
Resources:
  myLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Name: my-alb
      Type: application
      Scheme: internal
      Subnets:
        - !Ref subnet-AZ1
        - !Ref subnet-AZ2
      SecurityGroups:
        - !Ref mySecurityGroup
      LoadBalancerAttributes:
        - Key: "health_check_logs.s3.enabled"
          Value: "true"
        - Key: "health_check_logs.s3.bucket"
          Value: "amzn-s3-demo-logging-bucket"
        - Key: "health_check_logs.s3.prefix"
          Value: "logging-prefix"
```

Step 4: Verify bucket permissions

After health check logs are enabled for your load balancer, ELB validates the S3 bucket and creates a test file to ensure that the bucket policy specifies the required permissions. You can use the Amazon S3 console to verify that the test file was created. The test file is not an actual health check log file; it doesn't contain example records.

To verify that ELB created a test file in your S3 bucket

1. Open the Amazon S3 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/s3/.
2. Select the name of the bucket that you specified for health check logs.
3. Navigate to the test file, `ELBHealthCheckLogTestFile`. The location depends on whether you're using a prefix.
   - Location with a prefix: *amzn-s3-demo-logging-bucket*/*prefix*/AWSLogs/*123456789012*/ELBHealthCheckLogTestFile
   - Location without a prefix: *amzn-s3-demo-logging-bucket*/AWSLogs/*123456789012*/ELBHealthCheckLogTestFile

Troubleshooting

If you receive an access denied error, the following are possible causes:

- The bucket policy does not grant ELB permission to write health check logs to the bucket. Verify that you are using the correct bucket policy for the Region. Verify that the resource ARN uses the same bucket name that you specified when you enabled health check logs. Verify that the resource ARN does not include a prefix if you did not specify a prefix when you enabled health check logs.
- The bucket uses an unsupported server-side encryption option. The bucket must use Amazon S3-managed keys (SSE-S3).

Disable health check logs for your Application Load Balancer

You can disable health check logs for your load balancer at any time. After you disable health check logs, your health check logs remain in your S3 bucket until you delete them. For more information, see Creating, configuring, and working with buckets in the *Amazon S3 User Guide*.

Console

To disable health check logs

1. Open the Amazon EC2 console at https://eusc-de-east-1.console.amazonaws-eusc.eu/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 **Health check logs**.
6. Choose **Save changes**.

AWS CLI

To disable health check logs

Use the modify-load-balancer-attributes command.

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

Request tracing for your Application Load Balancer

When the load balancer receives a request from a client, it adds or updates the **X-Amzn-Trace-Id** header before sending the request to the target. Any services or applications between the load balancer and the target can also add or update this header.

You can use request tracing to track HTTP requests from clients to targets or other services. If you enable access logs, the contents of the **X-Amzn-Trace-Id** header are logged. For more information, see Access logs for your Application Load Balancer.

Syntax

The **X-Amzn-Trace-Id** header contains fields with the following format:

```
Field=version-time-id
```

Field

The name of the field. The supported values are `Root` and `Self`.

An application can add arbitrary fields for its own purposes. The load balancer preserves these fields but does not use them.

version

The version number. This value is 1.

time

The epoch time, in seconds. This value is 8 hexadecimal digits long.

id

The trace identifier. This value is 24 hexadecimal digits.

Examples

If the **X-Amzn-Trace-Id** header is not present on an incoming request, the load balancer generates a header with a `Root` field and forwards the request. For example:

```
X-Amzn-Trace-Id: Root=1-67891233-abcdef012345678912345678
```

If the **X-Amzn-Trace-Id** header is present and has a `Root` field, the load balancer inserts a `Self` field and forwards the request. For example:

```
X-Amzn-Trace-Id: Self=1-67891233-12456789abcdef012345678;Root=1-67891233-
abcdef012345678912345678
```

If an application adds a header with a `Root` field and a custom field, the load balancer preserves both fields, inserts a `Self` field, and forwards the request:

```
X-Amzn-Trace-Id: Self=1-67891233-12456789abcdef012345678;Root=1-67891233-
abcdef012345678912345678;CalledFrom=app
```

If the **X-Amzn-Trace-Id** header is present and has a `Self` field, the load balancer updates the value of the `Self` field.

Limitations

- The load balancer updates the header when it receives an incoming request, not when it receives a response.
- If the HTTP headers are greater than 7 KB, the load balancer rewrites the **X-Amzn-Trace-Id** header with a `Root` field.
- With WebSockets, you can trace only until the upgrade request is successful.

Troubleshoot your Application Load Balancers

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

Issues

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 Application Load Balancer target groups.

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

A security group does not allow traffic

The security group associated with an instance must allow traffic from the load balancer using the health check port and health check protocol. You can add a rule to the instance security group to allow all traffic from the load balancer security group. Also, the security group for your load balancer must allow traffic to the instances.

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

The network ACL associated with the subnets for your instances must allow inbound traffic on the health check port and outbound traffic on the ephemeral ports (1024-65535). The network ACL associated with the subnets for your load balancer nodes must allow inbound traffic on the ephemeral ports and outbound traffic on the health check and ephemeral ports.

The ping path does not exist

Create a target page for the health check and specify its path as the ping path.

The connection times out

First, verify that you can connect to the target directly from within the network using the private IP address of the target and the health check protocol. If you can't connect, check whether the instance is over-utilized, and add more targets to your target group if it is too busy to respond. If you can connect, it is possible that the target page is not responding before the health check timeout period. Choose a simpler target page for the health check or adjust the health check settings.

The target did not return a successful response code

By default, the success code is 200, but you can optionally specify additional success codes when you configure health checks. Confirm the success codes that the load balancer is expecting and that your application is configured to return these codes on success.

The target response code was malformed or there was an error connecting to the target

Verify that your application responds to the load balancer's health check requests. Some applications require additional configuration to respond to health checks, such as a virtual host configuration to respond to the HTTP host header sent by the load balancer. The host header value contains the private IP address of the target, followed by the health check port when not using a default port. If the target uses a default health check port, the host header value contains only the private IP address of the target. For example, if your target's private IP address is `10.0.0.10` and it's health check port is `8080`, the HTTP Host header sent by the load balancer in health checks is `Host: 10.0.0.10:8080`. If your target's private IP address is `10.0.0.10` and it's health check port is `80` then the HTTP Host header sent by the load balancer in health checks is `Host: 10.0.0.10`. A virtual host configuration to respond to that host, or a default configuration, may be required to successfully health check your application. Health check requests have the following attributes: the `User-Agent` is set to `ELB-HealthChecker/2.0`, the line terminator for message-header fields is the sequence CRLF, and the header terminates at the first empty line followed by a CRLF.

Clients cannot connect to an internet-facing load balancer

If the load balancer is not responding to requests, check for the following issues:

Your internet-facing load balancer is attached to a private subnet

You must specify public subnets for your load balancer. A public subnet has a route to the Internet Gateway for your virtual private cloud (VPC).

A security group or network ACL does not allow traffic

The security group for the load balancer and any network ACLs for the load balancer subnets must allow inbound traffic from the clients and outbound traffic to the clients on the listener ports.

Requests sent to a custom domain aren't received by the load balancer

If the load balancer is not receiving requests sent to a custom domain, check for the following issues:

The custom domain name does not resolve to the load balancer IP address

- Confirm what IP address the custom domain name resolves to using a command line interface.
  - **Linux, macOS, or Unix** – You can use the `dig` command within Terminal. Ex.`dig example.com`
  - **Windows** – You can use the `nslookup` command within Command Prompt. Ex.`nslookup example.com`
- Confirm what IP address the load balancers DNS name resolves to using a command line interface.
- Compare the results of the two outputs. The IP addresses must match.

If using Route 53 to host your custom domain, see My domain is unavailable on the internet in the *Amazon Route 53 Developer Guide*.

HTTPS requests sent to the load balancer return "NET::ERR_CERT_COMMON_NAME_INVALID"

If HTTPS requests are receiving `NET::ERR_CERT_COMMON_NAME_INVALID` from the load balancer, check the following possible causes:

- The domain name used in the HTTPS request does not match the alternate name specified in the listeners associated ACM certificate.
- The load balancers default DNS name is being used. The default DNS name cannot be used to make HTTPS requests as a public certificate cannot be requested for the `*.amazonaws.com` domain.

Load balancer shows elevated processing times

The load balancer counts processing times differently based on configuration.

- If AWS WAF is associated with your Application Load Balancer and a client sends an HTTP POST request, the time to send the data for POST requests is reflected in the `request_processing_time` field in the load balancer access logs. This behavior is expected for HTTP POST requests.
- If AWS WAF is not associated with your Application Load Balancer and a client sends an HTTP POST request, the time to send the data for POST requests is reflected in the `target_processing_time` field in the load balancer access logs. This behavior is expected for HTTP POST requests.

The load balancer sends a response code of 000

With HTTP/2 connections, if the number of requests served through one connection exceeds 10,000, the load balancer sends a GOAWAY frame and closes the connection with a TCP FIN.

The load balancer generates an HTTP error

The following HTTP errors are generated by the load balancer. The load balancer sends the HTTP code to the client, saves the request to the access log, and increments the `HTTPCode_ELB_4XX_Count` or `HTTPCode_ELB_5XX_Count` metric.

Errors

HTTP 400: Bad request

Possible causes:

- The client sent a malformed request that does not meet the HTTP specification.
- The request header exceeded 16 K per request line, 16 K per single header, or 64 K for the entire request header.
- The client closed the connection before sending the full request body.

HTTP 401: Unauthorized

You configured a listener rule to authenticate users, but one of the following is true:

- You configured `OnUnauthenticatedRequest` to deny unauthenticated users or the IdP denied access.
- The size of the claims returned by the IdP exceeded the maximum size supported by the load balancer.
- A client submitted an HTTP/1.0 request without a host header, and the load balancer was unable to generate a redirect URL.
- The requested scope doesn't return an ID token.
- You don't complete the login process before the client login timeout expires. For more information see, Client login timeout.
- The JWT authentication failed due to one of the following reasons:
  - The request is missing the Authorization header. (JWTHeaderNotPresent)
  - The token format in the request is invalid. This can occur when:
    - Token is malformed or missing mandatory parts (header, payload, or signature)
    - Header lacks the "Bearer" prefix
    - Header contains a different authentication type (e.g., "Basic")
    - Authorization header exists but token is missing
    - Multiple tokens are present in the request (JWTRequestFormatInvalid)
  - The token signature validation failed. This can occur when:
    - Signature doesn't match
    - Public key is invalid or cannot be converted to a decoding key
    - Public key size is not 2K
    - Token is signed with an unsupported algorithm
    - KID in the token is not present in the JWKS endpoint (JWTSignatureValidationFailed)
  - The JWT is missing a required claim for validation. (JWTClaimNotPresent)
  - The format of a claim's value in the JWT doesn't match the specified configuration format. (JWTClaimFormatInvalid)

HTTP 403: Forbidden

You configured an AWS WAF web access control list (web ACL) to monitor requests to your Application Load Balancer and it blocked a request.

HTTP 405: Method not allowed

The client used the TRACE method, which is not supported by Application Load Balancers.

HTTP 408: Request timeout

The client did not send data before the idle timeout period expired. Sending a TCP keep-alive does not prevent this timeout. Send at least 1 byte of data before each idle timeout period elapses. Increase the length of the idle timeout period as needed.

HTTP 413: Payload too large

Possible causes:

- The target is a Lambda function and the request body exceeds 1 MB.
- The request header exceeded 16 K per request line, 16 K per single header, or 64 K for the entire request header.

HTTP 414: URI too long

The request URL or query string parameters are too large.

HTTP 460

The load balancer received a request from a client, but the client closed the connection with the load balancer before the idle timeout period elapsed.

Check whether the client timeout period is greater than the idle timeout period for the load balancer. Ensure that your target provides a response to the client before the client timeout period elapses, or increase the client timeout period to match the load balancer idle timeout, if the client supports this.

HTTP 463

The load balancer received an **X-Forwarded-For** request header with too many IP addresses. The upper limit for IP addresses is 30.

HTTP 464

The load balancer received an incoming request protocol that is incompatible with the version config of the target group protocol.

Possible causes:

- The request protocol is an HTTP/1.1, while the target group protocol version is a gRPC or HTTP/2.
- The request protocol is a gRPC, while the target group protocol version is an HTTP/1.1.
- The request protocol is an HTTP/2 and the request is not POST, while target group protocol version is a gRPC.

HTTP 500: Internal server error

Possible causes:

- You configured an AWS WAF web access control list (web ACL) and there was an error executing the web ACL rules.
- The load balancer is unable to communicate with the IdP token endpoint or the IdP user info endpoint.
  - Verify that the IdP's DNS is publicly resolvable.
  - Verify that the security groups for your load balancer and the network ACLs for your VPC allow outbound access to these endpoints.
  - Verify that your VPC has internet access. If you have an internal-facing load balancer, use a NAT gateway to enable internet access.
- The user claim received from the IdP is greater than 11KB in size.
- The IdP token endpoint or the IdP user info endpoint is taking longer than 5 seconds to respond.
- The load balancer is unable to communicate with the JWKS endpoint, or the JWKS endpoint is not responding within 5 seconds.

- The size of the response returned by the JWKS endpoint exceeds 150KB or the number of keys returned by the JWKS endpoint exceeds 10
- The target group has target optimizer enabled and the agent encountered an unexpected error. See the section called “Troubleshoot target optimizer”.

HTTP 501: Not implemented

Possible causes:

- The load balancer received a **Transfer-Encoding** header with an unsupported value. The supported values for **Transfer-Encoding** are `chunked` and `identity`. As an alternative, you can use the **Content-Encoding** header.
- A websocket request was routed to a target group with target optimizer enabled.

HTTP 502: Bad gateway

Possible causes:

- The load balancer received a TCP RST from the target when attempting to establish a connection.
- The load balancer received an unexpected response from the target, such as "ICMP Destination unreachable (Host unreachable)", when attempting to establish a connection. Check whether traffic is allowed from the load balancer subnets to the targets on the target port.
- The target closed the connection with a TCP RST or a TCP FIN while the load balancer had an outstanding request to the target. Check whether the keep-alive duration of the target is shorter than the idle timeout value of the load balancer.
- The target response is malformed or contains HTTP headers that are not valid.
- The target response header exceeded 32 K for the entire response header.
- The deregistration delay period elapsed for a request being handled by a target that was deregistered. Increase the delay period so that lengthy operations can complete.
- The target is a Lambda function and the response body exceeds 1 MB.
- The target is a Lambda function that did not respond before its configured timeout was reached.
- The target is a Lambda function that returned an error or the function was throttled by the Lambda service.

- The load balancer encountered an SSL handshake error when connecting to a target.

For more information see How do I troubleshoot Application Load Balancer HTTP 502 errors in the AWS Support Knowledge Center.

HTTP 503: Service unavailable

Possible causes:

- The target groups for the load balancer have no registered targets, or all of the registered targets are in an `unused` state.
- The request was routed to a target group with target optimizer enabled, and was rejected because no targets were ready to receive requests. See the section called “Troubleshoot target optimizer”.

HTTP 504: Gateway timeout

Possible causes:

- The load balancer failed to establish a connection to the target before the connection timeout expired (10 seconds).
- The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.
- The network ACL for the subnet did not allow traffic from the targets to the load balancer nodes on the ephemeral ports (1024-65535).
- The target returns a content-length header that is larger than the entity body. The load balancer timed out waiting for the missing bytes.
- The target is a Lambda function and the Lambda service did not respond before the connection timeout expired.
- The load balancer encountered an SSL handshake timeout (10 seconds) when connecting to a target.

HTTP 505: Version not supported

The load balancer received an unexpected HTTP version request. For example, the load balancer established an HTTP/1 connection but received an HTTP/2 request.

HTTP 507: Insufficient Storage

The redirect URL is too long.

HTTP 561: Unauthorized

You configured a listener rule to authenticate users, but the IdP returned an error code when authenticating the user. Check your access logs for the related error reason code.

HTTP 562: JWKS Request Failed

The load balancer failed to receive a successful response from the JWKS (JSON Web Key Set) endpoint. A successful response should have a status code in the 200-299 range, but a different status code was received instead.

A target generates an HTTP error

The load balancer forwards valid HTTP responses from targets to the client, including HTTP errors. The HTTP errors generated by a target are recorded in the `HTTPCode_Target_4XX_Count` and `HTTPCode_Target_5XX_Count` metrics.

An AWS Certificate Manager certificate is not available for use

When deciding to use an HTTPS listener with your Application Load Balancer, AWS Certificate Manager requires you to validate domain ownership before issuing a certificate. If this step is missed during setup, the certificate remains in the `Pending Validation` state, and not available for use until validated.

- If using email validation, see Email validation in the *AWS Certificate Manager User Guide*.
- If using DNS validation, see DNS validation in the *AWS Certificate Manager User Guide*.

Multi-Line headers are not supported

Application Load Balancers do not support multi-line headers, including the `message/http` media type header. When a multi-line header is provided the Application Load Balancer appends a colon character, "`:`", before passing it to the target.

Troubleshoot unhealthy targets using the resource map

If your Application 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 Application 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 Application Load Balancer.

> **Note**
>
> You must enable **Show resource details** 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 configuration of the target group. Select a target groups 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 and confirm that applications running on the target are available. Select a targets ID to open its detail page in a new tab.

Selecting **Export** gives you the option of exporting the current view of your Application 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: HTTP Response Mismatch**
  - Verify the application running on the target is sending the correct HTTP response to the Application Load Balancer's health check requests.
  - Alternatively, you can update the Application Load Balancer's health check request to match the response from the application running on the target.

- **Unhealthy: Request timed out**
  - Verify the security groups and network access control lists (ACL) associated with your targets and Application Load Balancer are not blocking connectivity.
  - Verify the target has sufficient resources available to accept connections from the Application Load Balancer.
  - Verify the status of any applications running on the target.
  - The Application 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 status of any applications running on the target.
  - Verify the target is listening for traffic on the health check port.

    When using an HTTPS 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 any of your listeners have:

    - **FIPS post-quantum TLS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-FIPS-PQ-2025-09`
    - **FIPS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04`
    - **Post-quantum TLS policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-PQ-2025-09`
    - **TLS 1.3 policy** - Backend connections use `ELBSecurityPolicy-TLS13-1-0-2021-06`
    - All other TLS policies backend connections use `ELBSecurityPolicy-2016-08`

    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 Application Load Balancer to establish TLS handshakes.

Troubleshoot target optimizer

For detailed monitoring, see Target optimizer metrics

Configuration Errors

- `HTTPCode_ELB_502_Count`: The load balancer received a TCP RST from the agent when attempting to establish a connection.
- `HTTPCode_ELB_504_Count`: The load balancer failed to establish a connection to the agent before the idle timeout period elapsed.
- `HTTPCode_Target_5XX_Count`: The agent received a TCP RST from the target application when attempting to establish a connection. *(Applicable only when the target application itself is not generating this error response.)*

To fix these issues, please ensure that:

- The security groups on the targets are configured correctly.
- The agent is running with the expected configuration.
- The target application is running and listening on the TARGET_CONTROL_DESTINATION_ADDRESS configured in the agent.

Service Unavailable Errors (`HTTPCode_ELB_503_Count`)

Consistent HTTP 503 errors means that there are insufficient targets ready to receive requests from the ALB. The TargetControlRequestRejectCount metric is representative of these rejected requests. The TargetControlWorkQueueLength metric will fall to near zero values. To fix this issue, consider:

- Increasing the number of targets
- Setting the TARGET_CONTROL_MAX_CONCURRENCY variable on the agent to a larger value.

Health-check errors

- If the health check port is the same as TARGET_CONTROL_DATA_ADDRESS, then health check requests from the ALB are sent to the target application through the agent. If health checks are failing (due to HTTP 502 or Timeouts) refer to the Configuration Errors section.

Quotas for your Application 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 Application 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 ELB.

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 balancers
- Target groups
- Rules
- Trust stores
- Certificates
- HTTP headers
- Load Balancer Capacity Units

Load balancers

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

Name	Default	Adjustable
Application Load Balancers per Region	50	Yes
Certificates per Application Load Balancer (excluding default certificates)	25	Yes
Listeners per Application Load Balancer	50	Yes

Name	Default	Adjustable
Target Groups per Action per Application Load Balancer	5	No
Target Groups per Application Load Balancer	100	No
Targets per Application Load Balancer	1,000	Yes

Target groups

The following quotas are for target groups.

Name	Default	Adjustable
Target Groups per Region	3,000 *	Yes
Targets per Target Group per Region (instances or IP addresses)	1,000	Yes
Targets per Target Group per Region (Lambda functions)	1	No
Load balancers per target group	1	No

* This quota is shared by Application Load Balancers and Network Load Balancers.

Rules

The following quotas are for rules.

Name	Default	Adjustable
Rules per Application Load Balancer (excluding default rules)	100	Yes
Condition Values per Rule	5	No

Name	Default	Adjustable
Condition Wildcards per Rule	6	No
Match evaluations per rule	5	No

Trust stores

The following quotas are for trust stores.

Name	Default	Adjustable
Trust stores per account	20	Yes
Number of listeners using mTLS in verify mode, per load balancer.	2	No

Certificates

The following quotas apply to certificates, including advertising CA certificate names and certificate revocation lists.

Name	Default	Adjustable
CA certificate size	16 KB	No
CA certificates per trust store	25	Yes
CA certificates subject size per trust store	10,000	Yes
Maximum certificate chain depth	4	No
Revocation entries per trust store	500,000	Yes
Revocation list file size	50 MB	No
Revocation lists per trust store	30	Yes

Name	Default	Adjustable
TLS message size	64 K	No

HTTP headers

The following are the size limits for HTTP headers.

Name	Default	Adjustable
Request line	16 K	No
Single header	16 K	No
Entire response header	32 K	No
Entire request header	64 K	No

Load Balancer Capacity Units

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

Name	Default	Adjustable
Reserved Application Load Balancer Capacity Units (LCUs) per Application Load Balancer	15,000	Yes
Reserved Application Load Balancer Capacity Units (LCU) per Region	0	Yes

Document history for Application Load Balancers

The following table describes the releases for Application Load Balancers.

Change	Description	Date
Access token validation	This release adds support for the load balancer to validate JSON Web Tokens (JWT) provided by clients for secure service-to-service (S2S) or machine-to-machine (M2M) communications.	November 21, 2025
Transforms	This release adds support to transform host headers and URLs for incoming requests before the load balancer routes the traffic to a target.	October 15, 2025
Bucket policies for access logs and connection logs	Prior to this release, the bucket policy that you used depended on whether the Region was available before or after August 2022. With this release, the newer bucket policy is supported in all Regions. Note that the legacy bucket policy is still supported.	September 10, 2025
HTTP header modification	This release adds support for HTTP header modification for all response codes. Previousl y, this feature was limited to response codes 2xx and 3xx.	February 28, 2025

Capacity Unit reservation	This release adds support to set a minimum capacity for your load balancer.	November 20, 2024
Resource map	This release adds support to view your load balancer resources and relationships in a visual format.	March 8, 2024
One click WAF	This release adds support for configuring the behavior of your load balancer if it integrates with one click AWS WAF.	February 6, 2024
Mutual TLS	This release adds support for mutual TLS authentication.	November 26, 2023
Automatic Target Weights	This release adds support for the automatic target weights algorithm.	November 26, 2023
FIPS 140-3 TLS termination	This release adds security policies that use FIPS 140-3 cryptographic modules when terminating TLS connections.	November 20, 2023
Register targets using IPv6	This release adds support to register instances as targets when addressed by IPv6.	October 2, 2023
Security policies supporting TLS 1.3	This release adds support for TLS 1.3 predefined security policies.	March 22, 2023

Zonal shift	This release adds support to route traffic away from a single impaired Availability Zone through integration with the Amazon Application Recovery Controller (ARC).	November 28, 2022
Turn off cross-zone load balancing	This release adds support to turn off cross-zone load balancing.	November 28, 2022
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 28, 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 Application Load Balancers.	November 23, 2021
IPv6 internal load balancers	This release adds support to configure IPv6 target groups for Application Load Balancers.	November 23, 2021

AWS PrivateLink and static IP addresses	This release adds support to use AWS PrivateLink and expose static IP addresses by forwarding traffic directly from Network Load Balancers to Application Load Balancers.	September 27, 2021
Client port preservation	This release adds an attribute to preserve the source port that the client used to connect to the load balancer.	July 29, 2021
TLS headers	This release adds an attribute to indicate that the TLS headers, which contain information about the negotiated TLS version and cipher suite, are added to the client request before sending it to the target.	July 21, 2021
Additional ACM certificates	This release supports RSA certificates with 2048, 3072, and 4096-bit key lengths, and all ECDSA certificates.	July 14, 2021
Application-based stickiness	This release adds an application-based cookie to support sticky sessions for your load balancer.	February 8, 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

WAF fail open support	This release adds support for configuring the behavior of your load balancer if it integrates with AWS WAF.	November 13, 2020
gRPC and HTTP/2 support	This release adds support for gRPC workloads and end-to-end HTTP/2.	October 29, 2020
Outpost support	You can provision an Application Load Balancer on your AWS Outposts.	September 8, 2020
Desync mitigation mode	This release adds support for desync mitigation mode.	August 17, 2020
Least outstanding requests	This release adds support for the least outstanding requests algorithm.	November 25, 2019
Weighted target groups	This release adds support for forward actions with multiple target groups. Requests are distributed to these target groups based on the weight you specify for each target group.	November 19, 2019
New attribute	This release adds support for the routing.http.drop_invalid_header_fields.enabled attribute.	November 15, 2019
Security policies for FS	This release adds support for three additional predefined forward secrecy security policies.	October 8, 2019

Advanced request routing	This release adds support for additional condition types for your listener rules.	March 27, 2019
Lambda functions as a target	This release adds support for registering Lambda functions as a target.	November 29, 2018
Redirect actions	This release adds support for the load balancer to redirect requests to a different URL.	July 25, 2018
Fixed-response actions	This release adds support for the load balancer to return a custom HTTP response.	July 25, 2018
Security policies for FS and TLS 1.2	This release adds support for two additional predefined security policies.	June 6, 2018
User authentication	This release adds support for the load balancer to authentic ate users of your applications using their corporate or social identities before routing requests.	May 30, 2018
Resource-level permissions	This release adds support for resource-level permissions and tagging condition keys.	May 10, 2018
Slow start mode	This release adds support for slow start mode, which gradually increases the share of requests the load balancer sends to a newly registered target while it warms up.	March 24, 2018

SNI support	This release adds support for Server Name Indication (SNI).	October 10, 2017
IP addresses as targets	This release adds support for registering IP addresses as targets.	August 31, 2017
Host-based routing	This release adds support for routing requests based on the host names in the host header.	April 5, 2017
Security policies for TLS 1.1 and TLS 1.2	This release adds security policies for TLS 1.1 and TLS 1.2.	February 6, 2017
IPv6 support	This release adds support for IPv6 addresses.	January 25, 2017
Request tracing	This release adds support for request tracing.	November 22, 2016
Percentiles support for the TargetResponseTime metric	This release adds support for the new percentile statistic s supported by Amazon CloudWatch.	November 17, 2016
New load balancer type	This release of ELB introduces Application Load Balancers.	August 11, 2016